Bleek • Langlotz

Das große AMIGA 500 Buch

Das Referenzhandbuch für den Anwender, das alle Themen zum Amiga 500 ausführlich behandelt. Mit Praxis-Know-how zur Hardware, zur Programmierung und zur täglichen Arbeit.

DATA BECKER

Bleek • Langlotz

Das große AMIGA 500 Buch

DATA BECKER

Copyright © 1988 by DATA BECKER GmbH
Merowingerstr. 30
4000 Düsseldorf 1

3. überarbeitete Auflage 1990

Umschlaggestaltung Werner Leinhos

Text verarbeitet mit Word 5.0, Microsoft

Belichtumg Merlin Belichtungsstudio, Essen

Druck und buchbinderische Verarbeitung Mohndruck, Gütersloh

Alle Rechte vorbehalten. Kein Teil dieses Buches darf in irgendeiner Form (Druck, Fotokopie oder einem anderen Verfahren) ohne schriftliche Genehmigung der DATA BECKER GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

ISBN 3-89011-279-X

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle technischen Angaben und Programme in diesem Buch wurden von den Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler sind die Autoren jederzeit dankbar.

Vorwort

Der Amiga 500!

Dieser Computer vereint alles, was der Freak sich wünscht: die System-Software, die den Anwender mit dem versorgt, was er zum Arbeiten braucht, interessante Software für alle Anwendungsgebiete und natürlich auch Hardware, mit der man seinem Computer neue Fähigkeiten abverlangen kann.

Wir haben diesen Computer, den Amiga, näher untersucht und auch die Anwender, also Sie, näher betrachtet. Nach den Bedürfnissen und Erfahrungen, die von uns und unseren Bekannten festgestellt wurden, haben wir dieses Buch ausgerichtet. Die Arbeit von der Idee bis zum fertigen Fachbuch ist oft nicht so einfach, wie man sie sich vorstellt. Wir mußten planen, die Pläne verwerfen und neu gestalten, bis endlich ein Buchkonzept stand. Dieses Buchkonzept muß dann nur noch in Worte übersetzt werden. Dafür waren über fünf Monate aufwendiger Entwicklung notwendig, bei der nicht nur wir, die Autoren, die auf dem Cover erwähnt werden, mitgewirkt haben. Viele Menschen waren ein kleines Glied in der Kette. All diesen wollen wir in diesem Vorwort und auch in diesem Buch dafür danken, daß sie uns sehr unterstützt haben. Das fängt an bei den Bekannten in der täglichen Umgebung und zieht sich bis hin zu DATA BECKER in das Lektorat, das uns tatkräftig unterstützte.

Ein besonderer Dank geht an dieser Stelle an die Firmen, die uns, ohne zu zögern, ihre Produkte zur Verfügung stellten, um sie zu testen und in diesem Buch zu beschreiben. Gerade der Leser wird von der Bandbreite der Hard- und Software-Tests profitieren, denn diesen Erfahrungsüberblick wird man so schnell nie wieder bekommen. Unser letzter Dank geht an das letzte Glied in der Kette. An Sie! Ohne einen Leser wäre dieses Buch unnütz. Sie geben uns damit sozusagen den Sinn für unsere Arbeit.

Vielen Dank!

Großhansdorf,
im Juli 1990

Wolf-Gideon Bleek
Mirko Langlotz

Inhaltsverzeichnis

1. Einleitung 13

1.1 Der Reiseführer 14

2. Ich richte meinen Amiga persönlich ein 17

2.1 Die Startup-Sequence wird aufgeräumt 17
2.1.1 Endlich! Der günstigste Druckertreiber 26
2.1.2 Farben, die nicht auf die Augen gehen 30
2.2 Wir misten die Workbench aus 34
2.2.1 Was ist überhaupt auf der Workbench-Diskette? 35
2.2.2 Zahlensysteme 48
2.3 Sinnvolles Arbeiten im CLI 57
2.3.1 Geräte und Pfade 58
2.3.2 Kleine und praktische Tips 70
2.3.3 Die Selbstdarstellung der CLI-Befehle 76
2.3.4 Das Parameter-Format der CLI-Befehle 77
2.3.5 Dateinamen mit Sonderzeichen 78
2.3.6 Joker: Das DOS-Suchmuster 80
2.3.7 Abfragen im CLI 81
2.3.8 Disketten mit Auswahlmenü 84
2.3.9 Die richtige Arbeitsumgebung 87
2.3.10 Steuerzeichen im Text 89
2.3.11 Die unbekannte Sprache des Ed 94
2.3.12 Highlights des CLI 100
2.3.13 Wissenswertes über die Tastatur 112

3. Amiga-Software bringt Leben 123

3.1 Kleine Textverarbeitungen 124
3.2 Datenverwaltungen verschiedener Art 137
3.3 Grafikanwendung 145
3.4 Sound- und Klang-Anwendungen 162
3.5 Desk Top Publishing 176
3.6 Utilities 185

4. Hardware 193

4.1 Speichererweiterungen 194
4.1.1 Erweiterung auf 1 MByte 195
4.1.2 Manchmal nötig: Zerlegung des Amiga 200
4.1.3 Gigantisch - Aufrüstung auf 2.2 MByte 205
4.2 'ne Disk dazu 208
4.2.1 Die Alternative: 5¼-Zoll-Laufwerke 213
4.2.2 Anpassen der Software 216
4.3 Festplatten am Amiga 500 218
4.4 Soundsampling 242
4.5 Datenaustausch mit anderen Computern 251
4.5.1 Nullmodems 253
4.5.2 Anschluß und Anwendung 254
4.5.3 Ein externes CLI 258
4.6 Kleine Basteleien am kleinen Amiga 259
4.6.1 Kickstart-Umschaltung 259
4.6.2 Der preiswerte Fernseher 260
4.6.3 Schönheitsoperation am 1081 264
4.6.4 Mehr Rechenleistung für wenig Geld 266
4.6.5 Nullmodem im Eigenbau 268
4.6.6 Ein Schalter für den Speicher 269
4.6.7 Bau einer Joystick-Weiche 270
4.7 500er-Tuning 272

5. Vom Anwender zum Programmierer 275

5.1 Alles nutzen: Systemroutinen 279
5.1.1 Der Aufbau des Systems 279
5.1.2 Arbeiten mit den Amiga-Libraries 287
5.2 Programmieren in BASIC wie die Großen 318
5.2.1 Eine File-Request-Box 319
5.2.2 Ein Disk-Monitor 337
5.3 Mit GFA-BASIC zum Erfolg 357
5.3.1 GFA-BASIC stellt sich vor 357
5.3.2 Strukturiert programmieren leichtgemacht 361
5.3.3 Mit superschneller Grafik zum Erfolg 363
5.3.4 System-Routinen standardmäßig nutzen 366

6. Neue Welten - andere Programmiersprachen 379

6.1 Assembler 380
6.1.1 Einsteigen 382
6.1.2 Grundzüge der Sprache 387
6.1.3 Wie sag ich's ihm? - Die Adressierungsarten 400
6.1.4 Programmierung 405
6.1.5 Zugriff auf die Bibliotheken 416
6.1.6 Die richtige Eröffnung 423
6.1.7 Tempo ist angesagt: Primzahlen 425
6.2 C 433
6.2.1 Installation des Systems 434
6.2.2 Die ersten Schritte in C 440
6.2.3 Größere und kleinere Beispielprogramme 455

7. PC 465

7.1 Die Software-Lösung 466
7.2 Die Hardware-Lösung: SideCar oder PC-Karte 469
7.3 Arbeiten mit dem PC 470

8. Eine Geschichte über und zum Amiga 481

8.1 Was man in den Fingern hält: Die Hardware 482
8.2 Was den Amiga bedienbar macht: Die Workbench und das CLI 499
8.3 Was den Amiga steuert: Das Betriebssystem 502
8.3.1 Die verschiedenen Modulgruppen 504
8.3.2 Die Ergänzungen zum Grundstock 513

Anhang 517

A: Grundlagen und weiterführende Literatur 517
B: Fachwortlexikon für den Aufsteiger 518
C: Funktionen der Amiga-Libraries 541
D: Systemnahe BASIC-Befehle 552
E: Dankwort an die hilfeleistenden Firmen 558

Stichwortverzeichnis 563

1. Einleitung

Der Autor, das unbekannte Wesen. So sieht es für uns jedesmal aus, wenn wir ein Buch aufschlagen. Bei Bestseller-Autoren hört man viel über die Medien. Das Fernsehen berichtet, die Illustrierten wühlen in der Vergangenheit. Nicht so bei Fachbüchern. Hier stehen zwei Namen auf dem Umschlag, mehr nicht. Wir halten es deshalb für angebracht, daß Sie uns kennenlernen.

Mirko Langlotz ist intensiver Amiga-500-User. Er kam Ende 1988 in das DATA-BECKER-Autoren-Team und brachte sein umfangreiches Wissen auf den Gebieten Hardware und System-Erweiterungen mit. Dazu hat ihm nicht zuletzt die Kenntnis der Maschinensprache verholfen. Durch ein langjähriges Studium an den verschiedensten Rechnern baute er sich die Brücke in die PC-Welt.

Wolf-Gideon Bleek kam 1986 zu DATA BECKER. Mit einem Besuch auf der CeBit fing alles an, die ersten Demo-Texte, das erste Buch, weitere folgten wie auch dieses. Vorausgegangen waren autodidaktische Erfahrungen am ZX-81, einem Computer mit nur 1024 Bytes Speicher, am Commodore 64 (wie bei so vielen) und dem Amiga.

Wir beide sind, wie Sie hieran bestimmt erkennen können, ganz vernarrt in unsere Computer und bemühen uns, das Beste aus ihnen herauszuholen. Als wir uns trafen, dieses Buch zu entwickeln, besprachen wir unsere eigenen Probleme und diejenigen, die uns zu Ohren gekommen waren.

Denn gerade in der Anfangszeit, dem ersten Jahr, in dem man mit einem neuen Computer arbeitet, stellen sich viele Fragen, die häufig als Anfänger-Probleme bezeichnet werden. Diese Fragen werten den Anwender nicht ab! Aber sie sind typisch für ihn und erfordern viel Zeit, wenn man sie selbständig ohne fremde Hilfe lösen möchte.

Deshalb wendet man sich mit diesen Fragen und Problemen häufig an Freunde und Bekannte, die mehr Erfahrung besitzen. Entweder bringt dieser Bekannte die Geduld auf, zu jeder Frage eine ausführliche Anwort zu geben, oder aber man sitzt auf dem Trockenen und wird mit seinem Problem alleine gelassen. Um diese Situation zu vermeiden, haben wir unter anderem dieses

Buch geschrieben. Viele Fragen, die den Anfänger beschäftigen, haben wir in diesem Kapitel zusammengefaßt. Und wer ist schon so geduldig wie ein Buch? Immer wieder, wenn sich Ihnen eine Frage stellt, können Sie hier nachschlagen und die Antwort darauf nachlesen.

1.1 Der Reiseführer

Kommen wir jetzt zu einem kleinen Rundflug über die Kapitel. Nun, da alle Texte geschrieben sind, wollen wir Ihnen einen kurzen Überblick darüber geben, was wo zu finden ist. Das wird Ihnen das Zusammenleben mit diesem Buch wesentlich erleichtern. Bestimmt gibt es das eine oder andere Kapitel, das Sie brennend interessiert.

Damit Sie nun gleich dort nachlesen können, wo über ein Thema geschrieben wird, zu dem Sie schon immer brandheiße Informationen haben wollten, gehen wir jetzt die Kapitelinhalte locker durch.

Das erste Kapitel wurde von uns beiden geschrieben. In ihm versuchen wir, die wichtigsten Erfahrungen zu vermitteln, die wir in unserer Zeit mit dem Amiga gemacht haben.

Danach haben wir uns dann die Frage gestellt, wie es weitergehen soll, und sind zu dem Entschluß gekommen, daß die ersten Überlegungen zur fertigen Software und zu den Hardware-Erweiterungen gehen. Aus diesem Grund sind die Kapitel drei und vier entstanden. Nummer drei beschreibt das große Spektrum der Anwender-Software anhand von ausgewählten Beispielen. Zu allen vier wichtigen Gebieten wie der Text- und Datenverarbeitung, der Grafikanwendung und den Musikprogrammen findet der Leser mehrere Beispiele, die ihm deutlich machen, auf welche Punkte man bei der Auswahl seiner Software achten muß. Nicht die unermeßlichen Möglichkeiten stehen im Vordergrund, sondern die individuelle Anpassung an die täglichen Bedürfnisse machen die Software erst nutzbar.

Die Hardware-Landschaft ist im Vergleich zum Vorjahr um ein großes Maß gewachsen. Was gestern noch unmöglich oder zu teuer war, gibt es heute schon preiswert zu kaufen. Auch hier haben wir uns die Frage gestellt, was für den Amiga-500-Besitzer interessant sein könnte, und sind den Weg über die

altbekannten Erweiterungen bis hin zu ausgefallenen Hardware-Basteleien gegangen. Auch eine Festplatte zu einem erschwinglichen Preis haben wir getestet!

Der nächste Schritt im Leben eines 500er-Besitzers führt dann vom Anwender zum Programmierer, wie auch der zweite große Abschnitt unseres Buches betitelt ist. Die bereits mitgelieferte und leistungsstarke Programmiersprache AmigaBASIC ist das erste Thema. An dieser Stelle wollen wir uns aber nicht so sehr mit einer Einführung in diese Sprache begnügen. Dazu gibt es extra Bücher, die das in ausführlichster Weise tun. Unser Ziel war es, zu zeigen, wie man das Betriebssystem ansteuert. Durch das LIBRARY-Kommando können wir mit allen Funktionsbibliotheken arbeiten, und das machen wir dann auch. Das zweite gesteckte Ziel heißt dann, eine File-Request-Box zu programmieren, die alle Elemente des Betriebssystems benutzt.

Aber nicht nur die ziemlich einfache Interpreter-Sprache BASIC soll uns in diesem Zusammenhang beschäftigen. Sicher haben Sie schon einmal verträumt in die Richtung der "Großen" geschaut, die in Assembler oder C ein professionelles Programm schreiben. Wir beschreiten die Wege dorthin und werden Ihnen erklären, wie Sie sich einen Assembler zur Arbeit einrichten oder die Arbeitsdiskette zur C-Programmierung erstellen. Mit diesen soliden Voraussetzungen dürfte es kein Problem mehr sein, mit den ersten Beispielen den Erfolg in der Systemprogrammierung zu erhaschen. Wenn es Ihnen Spaß gemacht hat, überlegen Sie sich vielleicht, eine der beiden Sprachen zu lernen!

Im Amiga 500 steckt mehr, als man denken mag. Er läßt sich ohne weiteres als PC betreiben. Software- oder Hardware-Lösung heißt die oft gestellte Frage. Am Beispiel beider Lösungen wollen wir auch diesen Weg nicht unbeschritten lassen, um Ihnen auch hierzu einen Eindruck zu verschaffen.

Als Abschluß des Buches finden Sie einen Informations-Teil vor, der in seinem Umfang sicherlich einzigartig ist. Das Abschluß-Kapitel beschreibt in seinen 40 Seiten noch einmal übergreifend das Thema Amiga 500 mit seinen vielen tausend Fragen. Sie bekommen interne Informatioen zur Hardware, zur Benutzeroberfläche und zum Aufbau des Betriebssystems. Damit werden Zusammenhänge klar, die so allumfassend noch nie dargestellt wurden. Dazu haben wir für Sie in den Anhängen als Ergänzung ein Lexikon erstellt, in dem alle Fachbegriffe, die man in diesem Buch aufstöbern konnte, mit einfachen Worten erklärt werden.

2. Ich richte meinen Amiga persönlich ein

Der Amiga bietet im Gegensatz zu vielen älteren Computern die Möglichkeit, das gesamte System auf den Benutzer bezogen einzurichten. Dafür stellt er viele verschiedene Parameter zur Verfügung, die allesamt (in festgelegten Bereichen) verändert werden können.

Dem stehen aber meistens ahnungslose Benutzer gegenüber, die gar nicht darüber in Kenntnis gesetzt sind, daß Sie ihren Amiga in vielen Einstellungen ganz persönlich beeinflussen können.

Wir wollen uns deshalb im folgenden Text mit den über das System freigegebenen Parametern beschäftigen, die gerade das Individuelle eines Computersystems ausmachen und die wir ohne viel Aufsehen verändern können.

Ziel dieser Veränderung ist es, die persönliche Note des Benutzers auszudrücken, um so das Arbeitsklima um einiges zu verbessern. Denn der Arbeitende fühlt sich angenehm unterstützt und akzeptiert. Außerdem helfen viele Einstellungen bei der täglichen Arbeit und unterstützen so ein rasches Erledigen von Aufgaben, die auf anderen Amiga-Systemen viel weniger in Erscheinung treten. Gerade wegen dieser unterschiedlichen Nutzung ist es so wichtig, daß der Amiga von jedem Benutzer individuell eingerichtet wird.

2.1 Die Startup-Sequence wird aufgeräumt

Gehen wir für unsere Veränderungen vom Startvorgang des Amiga aus, der nach dem Einschalten abgewickelt wird. Die systeminternen Einstellungen werden ohne unsere Mitsprache ausgeführt. Der Benutzer kann an der Stelle das erste Mal eingreifen, an der die Startup-Sequence des CLI ausgeführt wird. Diese Sequenz ist aus CLI-Befehlen zusammengesetzt, die alle vor jeder anderen Arbeit ausgeführt werden. Sehen wir uns dazu eine fertige Startup-Sequence an, um anhand dieser eine eigene zu entwickeln. Ich habe dazu meine eigene genommen, die schon auf die Workbench 1.3 ausgerichtet ist. Ich werde aber an den entsprechenden Stellen gesondert darauf eingehen und hinweisen.

Hier ist die aktuelle Startup-Sequence:

```
c:run >NIL: c:SetPatch
cd c:
echo "A500/A2000 Workbench disk.  Release 1.3
Sys:System/FastMemFirst
BindDrivers
Addbuffers df0: 25
Addbuffers df1: 25
Addbuffers df2: 25
FF >NIL: -0
MakeDir RAM:T
Assign T: RAM:T
MakeDir RAM:Env
Assign ENV: RAM:Env
Path SYS:Utilities SYS:System RAM: SYS: S: add
SetMap D
Mount RAD:
Copy SYS:Disk.info RAM:
Copy SYS:Disk.info RAD:
SetClock >NIL: load
Resident CLI L:Shell-Seg SYSTEM pure
Resident c:Execute pure
Mount NEWCON:
LoadWB
EndCLI >NIL:
```

Ganz zu Anfang wird ein sogenanntes Patch-Programm geladen und im Hintergrund gestartet. Dieses befindet sich erst seit Version 1.3 auf der Workbench. Es ist für die Inhaber eines 68010-Prozessors gedacht, weil es eine Prozessor-Trap abfängt, die sonst zu einer Guru-Meditation führen würde. Auch die im System noch vorhandenen Fehler bei Speichererweiterungen bei $C000000 und der in der Funktion DeleteLayers() werden mit diesem kleinen Programm behoben. Es ist quasi ein das System überwachendes Programm, das die Kinderkrankheiten ausbügelt und seltene, aber ungewollte Fehler abfängt.

Allen, die sich eine eigene Startup-Sequence basteln wollen, kann man nur empfehlen, das Programm mit zu übernehmen, weil es nur wenig Speicher bzw. Rechnerzeit kostet und vor unnötigen Abstürzen schützt.

Das Kommando

```
cd C:
```

wurde an dieser Stelle nur gesetzt, um zu erreichen, daß alle Befehle, die nun aufgerufen werden, in diesem Verzeichnis gesucht werden. Liegt ein Befehl oder Programm irgendwo anders, so muß der vollständige Pfad angegeben werden. Dafür sehen wir auch gleich nach dem Echo-Kommando ein Beispiel. Das Programm FastMemFirst liegt im Verzeichnis System und muß deshalb mit der langen Pfad-Kennung versehen werden. Diese ließe sich ersparen, wenn man den Path-Befehl zuvor aufgerufen hätte.

Als nächstes folgt eine Sequence, die von mir in die Startup-Sequence eingebunden wurde. Das Kommando BindDrivers ergänzt in das System unter Umständen vorhandene Driver von zusätzlicher Hardware. Haben Sie eine nicht autokonfigurierende Speichererweiterung, so kümmert sich dieser Befehl darum, daß der Gerätetreiber aus dem Expansion-Verzeichnis gestartet wird, so daß ab jetzt der Speicher zur Verfügung steht. Das Kommando ist natürlich unnütz, wenn man keine Hardware-Erweiterungen besitzt, die erst konfiguriert werden müssen.

Die drei AddBuffers-Befehle ergänzen die Pufferspeicher für die drei angegebenen Diskettenlaufwerke. Man sollte bei genügend vorhandenem Speicher immer etwas mehr Pufferspeicher zur Verfügung stellen, damit der Zugriff flotter abgewickelt werden kann. Denn, was schon im Pufferspeicher liegt, braucht nicht erst von der Diskette geladen zu werden. So sind die Inhaltsverzeichnisse meistens bei mehr Speicher viel schneller griffbereit!

FF heißt mit vollem Namen FastFonts und hält, was es verspricht. Es steigert die Ausgabegeschwindigkeit der System-Zeichensätze um einiges. Damit werden Editor wie Ed oder die Shell schneller, und auch der AmigaBASIC-Editor kann so um einiges beschleunigt werden, so daß die Arbeit damit endlich wieder Spaß macht. Ich möchte diesen Komfort jetzt nicht mehr missen.

Die nächsten beiden Befehlsgruppen richten zwei Verzeichnisse im RAM ein und geben ihnen über die Assign-Kommandos gleichzeitig einen Device-Status. Das Verzeichnis T: wird für zeitlich begrenzte Zwischenspeicherungen von Texten und Daten verwendet. Es steht für englisch "temporary". Ich halte es allerdings für ungünstig, wenn man Backups im RAM ablegt, weil sie dort schneller verloren gehen könnten als auf der Diskette. Das ENV:-Verzeichnis ist für zwei neue CLI-Befehle eingeführt worden, die dort Zwischenspeicherungen von Umgebungs-Variablen vornehmen, welche Aussagen über den Zustand des Systems machen.

Path macht nun endlich das, was wir schon am Anfang gesagt hatten. Weitere Verzeichnisse werden als Pfade angegeben, um alle Befehle und Programme dem CLI zugänglich zu machen. Der Befehl wäre sicherlich am Anfang der Sequence sinnvoller gewesen, weil er dort einige Tipparbeit für den Verlauf erspart hätte.

```
setmap d
```

Endlich wird jetzt auch die deutsche Tastatur-Belegung aktiviert, damit besonders Anwender mit der Kenntnis des Maschinenschreibens ihre Fähigkeiten ausnutzen können. Schreiben Sie in einer anderen Sprache, z.B. Französisch oder Italienisch, dann ist es natürlich ratsam, eine andere Tastatur-Belegung zu wählen, vorausgesetzt, Sie beherrschen diese auch beim Maschinenschreiben. Auf jeden Fall ist dann aber gesichert, daß man auch die landesüblichen Sonderzeichen erreicht, die sonst nur schwerlich über die Alt-Tasten auf den Bildschirm zu bringen sind.

Der Mount-Befehl an sich birgt sicherlich keine Geheimnisse. Immer wieder interessant ist es aber, zu erfahren, was für ein Gerät über dieses Kommando neu in das System mit aufgenommen wird. In diesem Fall ist es die RAD:, eine reset-feste RAM-Disk, die sich auch auf der Workbench 1.3 befindet. Endlich hat man eine einigermaßen sichere Möglichkeit, schnell Daten zwischenzuspeichern, die selbst bei einer Guru-Meditation nicht verloren gehen.

Rein kosmetischer Art sind die beiden Copy-Kommandos, die das Icon der Workbench-Diskette auch für die RAM-Disk und die RAD: übernehmen. Dieses Kopieren hat die Aufgabe, auf der Workbench die drei Standard-Disketten in gleichem Aussehen erscheinen zu lassen, und ist auch nur sinnvoll, wenn es vor dem Aufruf von LoadWB durchgeführt wird. Denn sonst ist schon das Icon für die beiden RAM-Disks geladen. Man müßte dann über den Befehl DiskChange einen Diskettenwechsel vortäuschen, um die Workbench zum neuen Laden der Symbole zu veranlassen. Aber zerraufen wir uns nicht die Haare über diese umständlichen Vorgänge.

Wichtig wird es wieder beim Befehl SetClock, der aus der akku-gepufferten Uhr die Zeit ausliest und diese in die System-Uhr überträgt. Mit der Umlenkung der Ausgabe zum Gerät NIL: wird verhindert, daß im AmigaDOS-Window eine Textausgabe erscheint. Das macht den Vorgang eleganter und gleichzeitig schneller.

Der Resident-Befehl ist eine der ganz besonderen CLI-Befehle. Er erlaubt es, einen CLI-Befehl so in das RAM zu laden, daß er bei späteren Aufrufen nicht mehr von der Diskette oder aus dem Speicher geholt werden muß. Damit das auch wirklich funktioniert, wird hier in der Startup-Sequence zuerst einmal das dafür nötige Rüstzeug geladen. Dafür sind die beiden Befehlszeilen gedacht. In der zweiten wird dann schon gleich Execute in das System eingebunden. Der Execute-Befehl ist nun auch ohne die Workbench ansprechbar!

Wo wir uns jetzt allmählich dem Ende nähern, wird noch ein neues Gerät unter dem Namen NEWCON: eingebunden. Hiermit läßt sich ein Ein- und Ausgabe-Device ansprechen, das ähnlich wie das Console-Device (CON:) zur Editor-Unterstützung genutzt wird. Besonders das CLI machte bisher großen Gebrauch davon. Für die Verbesserungen, die am CLI-Editor durchgeführt wurden, reichte das alte Device aber nicht mehr aus. Deshalb entwarf man das New-Console-Device, das mit einer History-Funktion und anderen Leckerbissen glänzen kann.

Abschließend wird vor der Beendigung dieser Startup-Sequence die Workbench geladen. Das kann jetzt auch ohne Bedenken durchgeführt werden, denn die Icons sind von der System-Diskette kopiert, und auch alle anderen Veränderungen sind schon längst durchgeführt.

Nach dieser sehr ausführlichen Besprechung der Startup-Sequence wollen wir uns ansehen, welche Veränderungen ich im Laufe dieser Besprechung vorgenommen habe. Danach betrachten wir noch zusätzlich einige interessante Sonderfälle, die den einen oder anderen bestimmt interessieren werden.

```
C:Run >NIL: C:SetPatch
echo "A500/A2000 Workbench disk. Release 1.3
Sys:System/FastMemFirst
Addbuffers DF0: 25
Addbuffers DF1: 25
Addbuffers DF2: 25
FF >NIL:
Assign T: SYS:T
Path SYS:Utilities SYS:System RAM: SYS: S: add
SetMap d
Mount RAD:
Copy SYS:Disk.info RAM:
Copy SYS:Disk.info RAD:
SetClock >NIL:
Resident CLI L:Shell-Seg SYSTEM pure
Resident C:Execute pure
```

```
Mount NEWCON:
LoadWB
EndCLI >NIL:
```

In dieser zweiten Startup-Sequence werden Sie die gleichen Befehle wiederfinden wie in der ersten Version. Allerdings sind hier einige Vertreter entfernt worden, die in meinem persönlichen Fall unnötig waren. So hatte ich keine Hardware-Erweiterungen, weswegen das BindDrivers-Kommando überflüssig wurde. Der Cd-Befehl hatte sowieso keine Grundlage und konnte entfernt werden.

Da mir das Einrichten eines Verzeichnisses für temporäre Files im RAM ziemlich unnütz erschien, habe ich auch dafür die Befehle entfernt. Das Environment-Directory habe ich bei der Gelegenheit gleich mit gelöscht, weil es mir keine brauchbaren Dienste erweist. Hier heißt es aufmerksam sein, denn es kann sich zu einer hilfreichen Erweiterung entwickeln.

Sie sehen schon an diesen Kleinigkeiten, daß es nicht immer richtig ist, auf die Zusammenstellung einer Startup-Sequence zu vertrauen. In vielen Fällen wurde einfach zu viel des Guten getan, was dem Einzelnen nur negative Folgen bringt.

Denn jeder zuviel aufgerufene Befehl im Start kostet Zeit; einmal das Laden von der Diskette und zum anderen die Ausführzeit. Außerdem wird Speicherplatz auf der Diskette verschwendet, den man vielleicht gerade so nötig braucht.

Doch sehen wir uns jetzt die kleinen Kniffe an, mit denen wir auf ungewöhnliche Bedürfnisse reagieren können!

Wenn Sie noch nicht im Besitz einer Speicherkarte für Ihren Amiga 500 sind, werden Sie sicherlich auch die akku-gepufferte Uhr vermissen, die immer dafür sorgt, daß die System-Uhr richtig gestellt ist. Mit nur ein wenig Phantasie kann man sich hierbei helfen, so daß nicht immer Preferences geladen werden muß, um die aktuelle Uhrzeit zu stellen. Es gibt zwei Wege:

Der erste ist der einfachste: Wir gehen davon aus, daß das eigentlich Wichtige an der internen Uhr nur das Datum ist. Schließlich wird dieses gerade bei Files besonders gebraucht. Die Uhrzeit selbst ist zweitrangig. Denn alle Programme, wie z.B. Make bei der C-Programmierung, arbeiten ja auch mit

falschen Uhrzeiten. Kann man als zweite Voraussetzung noch festhalten, daß Sie Ihren Amiga nur einmal am Tag einschalten, so haben wir alles, was wir für unsere neue Startup-Sequence brauchen.

Ergänzen Sie einfach an einer beliebigen Stelle den CLI-Befehl

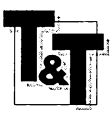
```
Date tomorrow
```

und schon wird immer beim Abarbeiten dieser Sequenz das Datum auf den nächsten Tag gestellt. Da es vorher immer auf dem des letzten Tages stand, ist das genau richtig. Der Amiga wird also bei jedem Booten mit dieser Befehlsfolge einen Tag älter.

Der andere Weg - hier ist es wichtig, daß man auch die Uhrzeit genau weiß - fordert etwas Arbeit vom Benutzer, indem er am Anfang bei der Startup-Sequence die Uhrzeit eingibt. Dieses erreicht man einfach durch das Schreiben von:

```
Date ?
```

Und schon fragt der Amiga nach der neuen Uhrzeit und dem neuen Datum. Sie sollten natürlich zuvor einen Text ausgeben, der den Benutzer darauf hinweist. Allerdings ist dies nur sinnvoll, wenn man auch eine Armbanduhr trägt oder über dem Computer bzw. im gleichen Raum eine Wanduhr hängt, denn ansonsten gerät man eher in Verlegenheit, als zur richtigen Uhrzeit zu kommen.



Ein besonderer Tip für diejenigen, die eine akku-gepufferte Uhr am Amiga haben; Sie werden diese ja sicherlich in der Startup-Sequence über das Kommando SetClock load richtig setzen. Haben Sie nun keine Uhr in der Nähe, weder am Arm noch an der Wand, so wäre es doch wünschenswert, wenn der Amiga diese Rolle übernehmen könnte. Das kann er auch! Sie müssen nur in der Startup-Sequence nach dem Laden der richtigen Uhrzeit die Clock von der Workbench aufrufen. Und zwar legen wir sie in einen separaten Task, so daß sie keinen stört. Als Parameter können wir jede beliebige Einstellung angeben, die auch von den Menüs der Uhr zugelassen sind. Hier ist eine Übersicht:

```
Clock-Parameter
Type:       ANALOG,  DIGITAL,  DIGITAL2
Mode:       12HOUR,  24HOUR
Special:    SECONDS, DATE
```

Auch die Größe des Windows kann mit =x,y,dx,dy angegeben werden.

Hier ein Beispiel:

```
Clock ANALOG=535,0,105,64 24HOUR SECONDS
```

Am Schluß der Startup-Sequence-Einrichtung möchte ich noch kurz ein Problem ansprechen, das vielen ein Dorn im Auge ist: der Zeitverbrauch. Bis die Startup-Sequence beendet, ist vergeht mindestens 1 Minute. Durch die große Anhäufung von CLI-Befehlen steht das Laufwerk nicht still und muß ständig einen Befehl nach dem anderen laden. Hierfür gibt es ein paar Tips, die den Zeitverbrauch heruntersetzen.

1. Haben Sie den Disketten-Puffer zu Anfang hoch eingestellt, sollten Sie gleiche Befehle hintereinander ausführen lassen. Dann muß der Befehl nämlich nicht nachgeladen werden, weil er sich noch im Pufferspeicher befindet - eine enorme Zeitersparnis.

2. Entfernen Sie alle unnötigen Befehle. Selbst wenn Sie Angst haben, daß der Befehl später doch einmal wichtig werden könnte, können Sie dies tun. Dann setzen Sie einfach ein Semikolon vor das Kommando. Dieses brauchen Sie später nur wieder zu entfernen und haben so keine Probleme, daß Ihnen vielleicht der Name oder das Format nicht mehr einfällt.

3. Für eine längere Textausgabe im Startup-Teil verwenden Sie um Himmels Willen keine ECHO-Kommandos. Zwar muß dieses bei großem Diskettenpuffer nicht nachgeladen werden, es ist aber trotzdem unschön. Es gibt einen viel einfacheren und schnelleren Weg. Tippen Sie den auszugebenden Text mit Ed in ein separates File, das Sie auch im S:-Verzeichnis ablegen. Dieses geben Sie dann später mit

   ```
   Type S:textfile
   ```

   wieder aus. Sie werden sehen, in welcher Geschwindigkeit dies abläuft!

Eine besondere Bremse der Startup-Sequence befand sich bei älteren Versionen im S:-Verzeichnis. Dort waren überaus gründliche Programmierer am Werk, die jeden Umstand erst überprüfen mußten. Das sieht dann so aus:

```
BindDrivers
echo "A2000  Workbench*N"
Assign WB: sys:
If EXISTS sys:s/JH0
   echo "Initializing Janus ...*N"
   Wait 25
   Echo "Mounting Janus Hard Disk*N"
   DJMount
   If EXISTS JH0:
      Assign WB: JH0:
      Echo "Transferring control to JH0: *N"
   Else
      Assign WB: sys:
   EndIF
EndIF
If EXISTS sys:s/DH0
   Assign WB: DH0:
   Echo "Transferring control to DH0: *N"
EndIF
AddBuffers WB: 30
CD WB:
If EXISTS WB:c
   Assign c: WB:c
EndIF
If EXISTS WB:s
   Assign s: WB:s
EndIF
If EXISTS WB:t
   Assign t: WB:t
EndIF
If EXISTS WB:l
   Assign l: WB:l
EndIF
If EXISTS WB:libs
   Assign libs: WB:libs
EndIF
If EXISTS WB:devs
   Assign devs: WB:devs
EndIF
If EXISTS WB:fonts
   Assign fonts: WB:fonts
EndIF
Assign sys: WB:
```

```
If EXISTS WB:System
   Assign System: WB:System
   Path add System
...
```

Die ersten beiden Blöcke, die Untersuchungen, ob Harddisks angesprochen sind, haben ihre Berechtigung, da man davon nicht ausgehen kann. Allerdings wäre hier eine feste Einstellung wesentlich sinnvoller, denn eine Harddisk tauscht man ja nicht jeden Tag aus. Vollkommen sinnlos ist allerdings der dann folgende Assign-Teil, bei dem jedes Verzeichnis auf seine Existenz geprüft wird. Sie können jetzt sicher gehen, daß die Startup-Sequence unter keinen Umständen abbrechen wird, aber Sie können genauso sicher sein, daß die Startup-Sequence ziemlich lange laufen wird.

Kürzt man alle Abfragen heraus, bleibt folgender Batch-Teil:

```
...
AddBuffers WB: 30
Assign c: WB:c
Assign s: WB:s
Assign t: WB:t
Assign l: WB:l
Assign libs: WB:libs
Assign devs: WB:devs
Assign fonts: WB:fonts
Assign sys: WB:
...
```

Sie sehen gerade an diesem Beispiel, daß strukturierte und äußerst korrekte Programmierung nicht unbedingt anwenderfreundlich ist, denn der Nutzen steht in keinem Verhältnis zum Schaden. Bei jedem Booten verliert man Zeit, und vielleicht wird ein Fehler nur einmal im Leben Ihres Amiga auftreten.

2.1.1 Endlich! Der günstigste Druckertreiber

Beim Kauf des Druckers spielen viele Faktoren eine Rolle. Sicherlich nicht unbeachtet bleibt auch der Preis. Deshalb wählt man einen preiswerten Drucker, der mit allen benötigten Funktionen aufwartet. Zu kurz kommt dann aber meistens der Blick darauf, ob und wie der Amiga überhaupt in der Lage ist, mit diesem Drucker zusammenzuarbeiten. Aber um dies festzustellen, gibt es einen einfachen Weg, den wir jetzt beschreiten werden:

Zuerst sei diesem Abschnitt vorausgeschickt, daß wir Ihnen raten möchten, einen Drucker, der den Epson- oder Commodore-Standard einhält, zu kaufen. Damit werden Sie nie Probleme haben. Es handelt sich dann nur noch um Feinheiten, die geklärt werden müssen. Eine zweite Möglichkeit sind alle Drucker, die innerhalb der Druckertreiber auf der Workbench- und der Preferences-Diskette angeboten werden. Die meisten Exemplare sind hier aufgeführt. Oder aber Sie haben einen Druckerverkäufer, der Ihnen den passenden Druckertreiber dazu liefert. Auch dann sollten sich keine Komplikationen ergeben.

Haben Sie sich nun Ihren persönlichen Drucker gekauft, sollten Sie diesen zuerst richtig anschließen. Dazu verbinden Sie den Ausgang des Computers mit dem dazugekauften Kabel, das andere Ende wird am Amiga eingesteckt. Und zwar entweder am Ausgang Port seriell oder Port parallel. Dies ist wichtig für die späteren Einstellungen im Programm Preferences.

Den Druckertreiber auswählen

Das ist gleich das nächste, was Sie tun müssen: Schalten Sie alle Geräte (also den Amiga, den Drucker und auch den Monitor) ein, und booten Sie die Workbench. Dort starten Sie das Programm Preferences und gehen auf die Seite mit den Druckereinstellungen.

Dort stellen Sie, je nachdem, ob der Drucker am seriellen oder parallelen Port angeschlossen ist, die Symbole in der linken oberen Ecke ein. Jetzt weiß der Amiga, wohin er die Daten schicken muß. Als nächstes sollten wir uns überlegen, welcher Druckertreiber der richtige ist. Dazu wählen Sie auf der rechten Seite über die beiden großen Pfeile den Druckertreiber, der z.B. den Namen Ihrer Druckerfirma und vielleicht sogar eine ähnliche oder die richtige Typenbezeichnung enthält. Ist der Druckertreiber nicht dabei, so kann es noch sein, daß wir auf der Extras-Diskette weitere finden.

Dann müssen Sie mit dem Programm Install-Printer arbeiten, das seit neuerer Zeit auf der Workbench-Diskette im Verzeichnis Utilities ist. Es gibt nach dem Start von der Workbench in einem neu geöffneten CLI-Fenster eine Liste aller Druckertreiber der Extras-Diskette aus, aus der man sich seinen gewünschten aussuchen kann. Dieser wird dann auf die Workbench-Diskette kopiert und kann über Preferences ausgewählt werden.

Den Druckertreiber testen

Haben Sie jetzt einen Treiber erwischt, der mit Ihrem Drucker zusammenarbeiten könnte, so geht es daran, Funktionen des Druckers ausführlich zu testen. Dafür steht uns leider kein spezielles Test-Programm zur Verfügung, so daß wir selbst die Funktionen des Druckers austesten müssen. Das läßt sich am besten mit dem Notepad machen.

Dazu geben Sie einen Text ein, der etwa das folgende Aussehen hat:

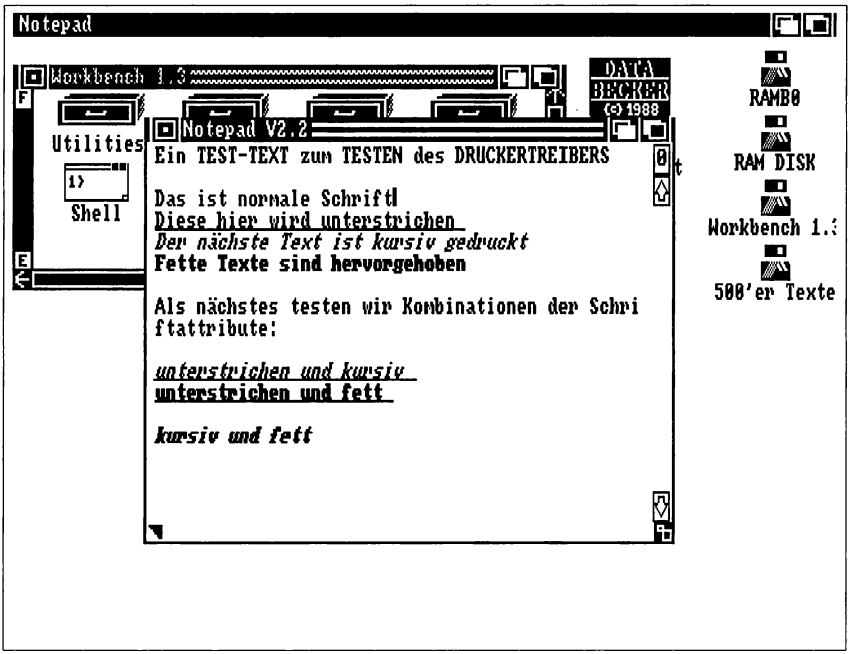


Abb. 2.1: Einen Test-Text mit dem Notepad kreieren

Hiermit testen wir zuerst die verschiedenen Schriftattribute wie unterstreichen, kursiv und fett. Sie können nach diesem Test herausfinden, ob der Druckertreiber überhaupt in der Lage ist, Texte auszugeben und ob diese auch mit den richtigen Attributen versehen werden. Dazu wählt man bei der Ausgabe vom Notepad Draft an, um in den Textmodus zu schalten. Bei der Funktion Print können Sie dann mit Auto-Size den Ausdruck starten.

Um auch die Grafikeigenschaften zu testen, wählen wir jetzt noch verschiedene Zeichensätze aus. Hier ist mein Vorschlag für einen Test-Text:

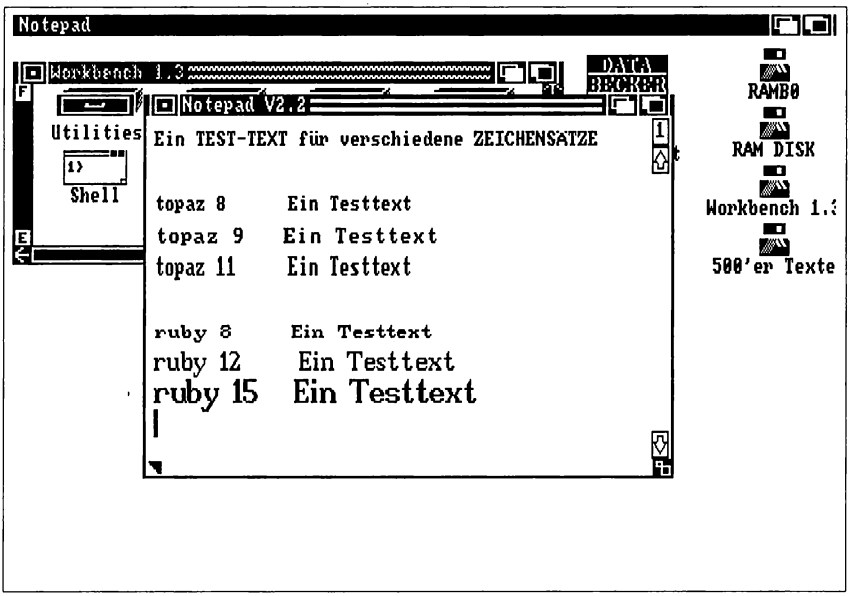


Abb. 2.2: Ein Test-Text für den Grafikausdruck

(Für diesen Beispieltext wählt man zuerst GLOBAL FONTS aus und ruft dann einen neuen Zeichensatz auf. Damit wird eine Zeile geschrieben und ein weiterer Zeichensatz ausgesucht. Das wiederholt man so lange, bis eine Seite vollgeschrieben wurde.)

Diesen Text drucken wir auch aus. Dafür stellen wir den Druck-Modus aber auf Graphic und wählen beim Ausdruck die Funktion Small, um nicht zu viel Platz auf einer Druckseite zu verbrauchen. Nun sollte es sich zeigen, ob der Druckertreiber die Grafik-Informationen richtig übersetzt oder ob er nur falsche Steuerzeichen sendet. Sie können es klar am Ausdruck erkennen.

Sollten nur wirre Folgen von irgendwelchen Symbolen, Texten oder Punktmustern gedruckt werden, so haben wir den falschen Druckertreiber. Es ist dann das Beste, den Drucker auszuschalten und zu warten, bis der System-Requester Printer-Trouble kommt. Diesen beantworten Sie mit Cancel und

wählen in Preferences erneut einen ähnlichen Druckertreiber. Hier wiederholt sich dann die Prozedur, bei der Sie immer zuerst einen normalen Text testen und dann erst die Grafik ausprobieren. Denn es ist im Normalfall unmöglich, daß ein Drucker nur im Grafikmodus funktioniert.

Eine andere durchaus gebräuchliche Möglichkeit, den Grafikausdruck zu testen, ist das Programm GraphicDump im Utitlities-Verzeichnis. Sie rufen es ganz einfach durch einen Doppelklick auf, und schon wird die gesamte Screen als Grafik zum Drucker geschickt. Es wird sich dann genau wie bei dem Grafikausdruck des Notepad-Textes zeigen, ob der Druckertreiber geeignet ist.

Sie werden bemerkt haben, daß die Auswahl des Druckertreibers keine einfache Aufgabe ist. Doch am Ende der Mühe steht das gute Ergebnis: ein voll funktionierender Text- und Grafikausdruck. Der gewählte Druckertreiber kann Sie immer begleiten. Sie können Ihn auf jede Workbench kopieren und sind so immer sicher, daß alle Programme Ihre Ergebnisse richtig zu Papier bringen.

Wer schon den für sich richtigen Druckertreiber auf einer älteren Workbench ausgesucht hat und meint, diesen am besten zu kopieren, damit er sich die Auswahl erspart, hat falsch gedacht. Bei jeder neuen Workbench-Version werden z.B. auch die Druckertreiber neu gestaltet. So sind die Treiber der Version 1.3 um mehr als 2000 % schneller in der Grafikausgabe. Es lohnt sich also, bei neuen Workbench-Versionen wiederum einen neuen Druckertreiber auszuwählen.

2.1.2 Farben, die nicht auf die Augen gehen

Unter allen Einstellungen, die auf der Workbench mit dem Programm Preferences gemacht werden können, sind die Farbeinstellungen die strittigsten. "Farbe ist Geschmackssache" sagt der Volksmund und hat damit gar nicht so unrecht. Aber es gibt nicht nur diesen einen Aspekt. Der andere - das ist der fachliche - hilft uns bei der Farbwahl, eine individuelle Harmonie herzustellen, die das Arbeiten auf dem Amiga angenehmer macht.

Aber tragen wir erst die Voraussetzungen zusammen, die uns von der Workbench geboten werden. Da haben wir zuerst die Screen. Sie hat 640 x 256

Bildpunkte. Darum ist ein Rand in der Hintergrundfarbe gelegt, der nicht beschrieben oder mit Grafik versehen werden kann. Neben der Hintergrundfarbe haben wir dann noch drei weitere Farbstifte, die für Schrift und Grafik verwendet werden können.

Alle Farben des Amiga werden aus den drei Grundfarben der Lichtmischung zusammengesetzt. Damit sind Rot, Grün und Blau gemeint, die auch beim Fernseher zur Erstellung des Fernsehbildes benutzt werden. Man bezeichnet dieses Mischverfahren als das "Additive Farbmischverfahren", weil wir durch Zusammenfügen der Grundfarben schließlich zur alles umfassenden Farbe kommen: Weiß.

Nebenbei: Das steht im Gegensatz zur subtraktiven Farbmischung, bei der durch das Mischen der Farben immer weniger Licht vorhanden ist und damit zum Schluß Schwarz erreicht wird. Aber damit brauchen wir uns nicht zu beschäftigen.

Zum Demonstrieren unserer Erklärungen möchte ich Sie jetzt bitten, das Programm Preferences zu starten. Sie finden es auf der Workbench-Diskette im Ordner Prefs. Sind wir nach einiger Ladezeit im Programm, sehen wir die Hauptseite vor uns, auf der auch die vier Workbench-Farben eingestellt werden können. Dazu finden sich im unteren Drittel vier Kästen, jeder für eine Farbe. Außerdem befinden sich noch weiter darunter drei Schieberegler, die mit R, G und B bezeichnet sind, analog zu Rot, Grün und Blau.

Von Commodore werden die Farben auf die folgenden vier Werte eingestellt: Die erste Farbe ist für den Hintergrund vorgesehen und besteht aus einem hellen, nicht sehr kräftigen Blau. Die zweite Farbe, die Textfarbe, ist ein volles Weiß, genau wie die dritte Farbe Schwarz ist. Die vierte und letzte Farbe ist zum besonderen Hervorheben gedacht. Es ist meistens ein Orange, das bei neueren Workbench-Versionen mehr zum Rot hin tendiert.

Möchte man diese Farben nun individuell und neu gestalten, muß man sich zuerst überlegen, welche Bedeutung dies für die späteren Darstellungen hat. Der Hintergrund bzw. die Hintergrundfarbe soll dem Text oder der Grafik eine gute Unterlage sein und diese deutlich erkennen lassen. Das heißt für uns, daß ein gewisser Kontrast herrschen muß, damit man die Abgrenzungen zwischen Hintergrund und Text leicht identifiziert. Andererseits darf der Unterschied nicht zu kraß sein, denn dann wird das Auge überanstrengt. Probieren Sie dies jetzt einmal aus:

Stellen Sie dafür im Programm Preferences die Hintergrundfarbe auf Schwarz. Haben Sie jetzt einen mit weiß geschriebenen Text auf dem Bildschirm, ist dieser zwar noch ohne weiteres erkennbar, es strengt die Augen aber sehr an, diesen Text zu lesen. Das zeigt uns, daß der Kontrast zwischen Schrift und Hintergrund nicht zu scharf sein darf, damit für die Augen keine Belastungen entstehen. Das bezeichnet man als den Helligkeitskontrast, der für eine gute Lesbarkeit weder zu groß noch zu klein sein darf. Das richtige Mittelmaß ist gefragt.

Nun wollen wir den anderen Fall erproben, bei dem auf einen Kontrast verzichtet wird. Schieben Sie dazu alle Regler unter das r des Textes Reset Colors, der über den Schiebereglern steht. Sie werden unschwer erkennen, daß auch dieser Zustand die Texte und Zeichen, die in der weißen Schrift geschrieben sind, unweigerlich schwer lesbar macht, und auch hier treten Schmerzen in den Augen auf. Es muß also ein Zwischenweg zwischen starkem und schwachem Kontrast gefunden werden.

Es zeigen sich aber noch ganz andere Probleme bei der Zusammenstellung der eigenen Farben. Wir haben hier zuerst auf das Kontrast-Problem aufmerksam gemacht, das bei Hell/Dunkel-Unterschieden auftritt. Es gibt das gleiche Problem aber auch bei Farben.

Setzen Sie zum Testen dieses Verhaltens die Hintergrundfarbe auf ein sattes Grün, indem Sie nur den Regler des Grüntons aufschieben und alle anderen auf Null setzen. Gleichzeitig wählen Sie für die Schriftfarbe ein sattes Rot, indem Sie auf die gleiche Weise vorgehen.

Sie werden erkennen, daß auch diese beiden Farben nicht zusammen passen. Sie beißen sich! Das liegt daran, daß beide Töne auf dem Farbkreis auf einer waagerechten Linie liegen; sie verstärken sich gegenseitig, wodurch ein Flimmern an den Grenzen der Farbtöne entsteht. Das ist besonders bei Schrift, die ja nur aus Linien besteht, sehr ungünstig. Genau wie der Helligkeitskontrast ist auch der hier beschriebene Farbkontrast zu handhaben. Stehen sich die beiden Farbtöne klar gegenüber, ist der Kontrast zu groß. Werden sie aber zu stark verklärt, lassen sie sich nicht mehr unterscheiden, weil der Kontrast zu klein ist.

Mit dem, was sie an diesen Beispielen gelernt haben, lassen sich zwar keine richtigen Farbeinstellungen bestimmen, es reicht uns aber zum Setzen von Regeln für die Farbwahl.

Hier sind meine Vorschläge für die farbliche Gestaltung:

Der Hintergrund

Der Hintergrund sollte aus einer dunkleren Farbe, die nicht in ihrem klaren Ton gewählt ist, bestehen. Am besten wäre ein Braun- oder Blauton, der durch Mischen mit etwas Grün oder Rot leicht geschwächt wird.

Die Schrift

Für die Schrift muß eine Kontrastfarbe zum Hintergrund gewählt werden, damit sie später klar und deutlich erkennbar ist. Ist der Hintergrund dunkel gehalten, sollte die Schrift in einem hellen Farbton gewählt werden. Allerdings darf es kein scharfer Kontrast zum Hintergrund sein, damit die Konturen nicht zu flimmern beginnen.

Die Signalfarbe

Die dritte Farbe der Amiga-Workbench sollte als Signalton gewählt werden. Man benutzt sie häufig, um ein Warnblinken auszulösen oder wichtige Stellen zu markieren. Daran sollten Sie sich ganz unabhängig von der Farbgestaltung halten, damit auch andere Programme, die auf der Workbench laufen, einwandfrei funktionieren. Wenn Sie die anderen beiden Voraussetzungen für den Hintergrund und die Schriften halten, möchte ich hier einen grellen rötlichen oder gelblichen Farbton empfehlen. Er hebt sich besonders davon ab.

Die Blockfarbe

Der letzte Farbpinsel wird von Intuition bei den Windows nur im Depth-Arrangement-Gadget verwendet. Es sollte eine satte, unauffällige Farbe sein. Selbst wenn der Hintergrund schon mit einem dunklen Ton belegt ist, kann auch hier eine andere Farbe gewährt werden. Sie sollten dann nur in den Komplementärbereich wechseln, damit der Anwender die beiden Farben unterscheiden kann. Das heißt, speziell auf unseren Fall angewendet, daß Sie

für einen bräunlichen Hintergrund einen grünlichen oder gelblichen Farbton wählen müssen. Dieser muß aber nicht klar und strahlend sein, sondern kann mit den anderen Farben dunkel und "dreckig" gemischt werden.

Zum Schluß ...

Der Mauspfeil

Auch hier geht es um die Farbeinstellung. Gerade der Mauspfeil muß farblich besonders abgehoben eingestellt werden, damit er sofort gefunden und leicht erkannt werden kann. Deshalb sollten Sie die folgenden Regeln beachten.

Verwenden Sie für den Mauspfeil immer die Komplementärfarben. Für ein Blau im Hintergrund sollte der Mauspfeil rot gestaltet sein, wie es auch auf der Standard-Workbench der Fall ist. Wählen Sie Grün, ist die entgegengesetzte Farbe zwar auch Rot, sollte aber nicht in reiner Form genommen werden, da hier das bekannte Flimmern auftritt. Bei Gelb gibt es keine Probleme, denn hier ist Violett die gegenüberliegende Farbe.

Wichtig bei der Auswahl der Grafik für den Mauspfeil ist auch immer, daß man nicht nur eine Farbe verwendet, wie es manche Programme machen. Nehmen Sie ruhig eine Hauptfarbe, die Sie aber an den Rändern noch mit anderen - möglichst kontrastreichen - Farben abgrenzen. So hat man nämlich auch die Gewähr, daß beim Überfahren von Flächen einer Farbe nicht die unseres Mauspfeils im Gewühle ganz untergeht. Der Benutzer soll schließlich wissen, wo sein Pfeil ist!

2.2 Wir misten die Workbench aus

Ein schöner Titel! Die Workbench ist so, wie sie ausgeliefert wird, sicher nicht optimal ausgestattet. Wie Sie in einem vorhergehenden Kapitel gehört haben, sind einige Einstellungen vorgesehen und möglich, die noch vorgenommen werden müssen oder können.

Gerade hier liegt es an der Eigeninitiative, von der es abhängt, inwieweit man auf der Workbench Veränderungen vornimmt. Dieser Abschnitt des Kapitels über unsere Erfahrungen mit dem Amiga 500 soll besonders den Aspekt be-

leuchten, der sonst immer vernachlässigt wird. Wir werden hier über die "Intimitäten" der Workbench sprechen, über den Aufbau der Diskette, die für den Start und die Arbeit mit dem Amiga lebensnotwendig ist.

Was muß eigentlich auf der Workbench-Diskette sein? Und wozu? Welche Aufgaben erfüllen die einzelnen Verzeichnisse, Daten und Programme? Einer Frage nach der anderen werden wir jetzt im folgenden Text nachgehen, damit Sie am Ende erstens wissen, woraus ihre Workbench besteht, und zweitens, wie man sich aus den brauchbaren Dingen eine eigene zusammenstellt.

2.2.1 Was ist überhaupt auf der Workbench-Diskette?

Die Workbench-Diskette des Amiga 500 wird mit folgendem Verzeichnis ausgeliefert:

```
   c (dir)
   L (dir)
   S (dir)
   T (dir)
   libs (dir)
   devs (dir)
   fonts (dir)
   Prefs (dir)
   System (dir)
   Utilities (dir)
   Expansion (dir)
   Empty (dir)
   Trashcan (dir)
.info                          Clock
Clock.info                     Disk.info
Empty.info                     Expansion.info
Prefs.info                     Shell
Shell.info                     System.info
Trashcan.info                  Utilities.info
```

Wir wollen jetzt nacheinander die Verzeichnisse der Workbench-Diskette durchgehen, weil es hier naheliegt, anzunehmen, daß diese entscheidende Bedeutungen haben. Schließlich kommen auf beiden Disketten etwa die gleichen Verzeichnisse vor.

Die Verzeichnisse der Workbench-Diskette

Empty

Dieses Verzeichnis darf unter keinen Umständen persönliche Daten enthalten. Es ist einzig und allein dazu bestimmt, von der Workbench aus neue Verzeichnisse einzurichten. Dazu wird, das kennen Sie bestimmt schon, das Icon der Empty-Schublade angeklickt und die Funktion Duplicate aus dem Workbench-Menü angewählt. Danach haben wir ein zweites Verzeichnis, dessen Namen wir nach Belieben verändern können.

Damit dieser Vorgang aber ordnungsgemäß ablaufen kann, ist es nötig, daß die Schublade leer ist, denn sonst besäße die Kopie noch einmal diesen Inhalt, was sicherlich unnütz ist. Denn wir wollen ja eine neue, jungfräuliche, d.h. leere Schublade ohne Daten.

Trashcan

Bei der Trashcan handelt es sich um das Verzeichnis, in das alle Programme oder Schubladen geschoben werden, die man nicht mehr benötigt. Die Trashcan ist eine Art Abstellgleis. Mit der Workbench-Funktion Empty Trash können der Mülleimer geleert und die darin enthaltenen Daten gelöscht werden. Auch dieses Verzeichnis sollte man auf der Diskette lassen, denn ohne ein solches wären diese eben beschriebenen Operationen nicht möglich, und so sehr viel Platz nimmt das leere Verzeichnis nicht in Anspruch.

C:

Das C-Verzeichnis hat eine äußerst wichtige Bedeutung. Das sieht man schon daran, daß es neben seinem Namen c noch als virtuelles Device definiert ist. Das heißt, egal, wo die CLI-Befehle auch liegen, unter dem Namen C: findet man sie immer. Das ist auch sehr wichtig, denn einige der CLI-Befehle werden schon beim Einschalten des Amiga benötigt. Deswegen muß der Amiga ganz klar wissen, wo die gesuchten Kommandos zu finden sind. Allerdings ist nur der Ort vorgeschrieben. Was man dort finden kann, ist es nicht, und das nutzen wir aus:

Will man nämlich auf der Workbench-Diskette Platz schaffen oder einfach das Unbrauchbare ausmisten, dann finden sich hier sicher einige Befehle, die nicht mehr gebraucht werden. Zuerst sollten Sie aber bedenken, daß die CLI-Befehle permanent verbessert werden. So sitzen die Programmierer von Commodore ständig daran, die Befehle (eigentlich sind es kleine Programme) immer kürzer zu machen, damit sie weniger Speicherplatz beanspruchen. Man kann also schon einige KByte sparen, wenn man sich nur die neuesten Versionen besorgt. Außerdem befinden sich aber unter den Befehlen einige, die so gut wie nie verwendet werden. Weshalb sollte man dafür Speicherplatz verschwenden? Dieser läßt sich besser nutzen!

Sie sollten also jetzt Ihr CLI-Verzeichnis durchgehen und bei jedem Eintrag überlegen, ob der Befehl für Sie oder das System wichtig ist. Ist er das nicht, kann er gelöscht werden, ansonsten sollten Sie ihn behalten. Für das System ist ein Befehl wichtig, wenn er z.B. in der Startup-Sequence oder anderen Batch-Dateien steht. Für Sie ist ein Befehl notwendig, wenn Sie ihn auch nur einmal im Monat benutzen, denn dieses eine Mal könnte so viel Zeit dafür verbraucht werden, den Befehl von einer anderen Diskette zu laden, daß die Speicherersparnis wieder verloren ginge. Das wollen wir nicht! Als aktuelles Beispiel habe ich hier das C:-Verzeichnis der Workbench-Diskette Version 34.20 abgedruckt, in der auch alle Befehle der alten Versionen enthalten sind. Ich möchte einige Befehle herauspicken, die entfallen könnten:

```
 +AddBuffers              *Ask
 *Assign                  #Avail
 +Binddrivers             *Break
  CD                      *ChangeTaskPri
  Copy                     Date
  Delete                   Dir
 +DiskChange               DiskDoctor
  Echo                     Ed
 #Edit                    *Else
  EndCLI                  *EndIf
 *EndSkip                 *Eval
  Execute                  Failat
  Fault                    FF
 *FileNote                *GetEnv
  IconX                   *If
 *Info                    *Install
 *Join                    *Lab
  List                     LoadWB
:                                  :
```

```
:                                    :
 #Lock                               Makedir
  Mount                              NewCLI
  NewShell                           Path
 *Prompt                            *Protect
 *Quit                              *Relabel
 +RemRAD                             Rename
 +Resident                           Run
 *Search                            +SetClock
  SetDate                           *SetEnv
  SetPatch                          *Skip
 *Sort                              #Stack
 *Status                             Type
  Version                           *Wait
  Which                              Why
```

Legende:
+ nur bei viel Speicher/Hardware-Erweiterungen
* nur bei intensiver Arbeit im CLI
# als Hilfsmittel zum Programmieren

Die Legende soll Ihnen eine erste Orientierung ermöglichen. Sie kann aber nicht eine Erklärung ersetzen, die jetzt folgt: Manche CLI-Befehle kann man nur gebrauchen, wenn bestimmte Hardware-Erweiterungen vorhanden sind. So ergänzt BindDrivers die Hardware-Treiber von Zusatzgeräten und SetClock setzt die akku-gepufferte Uhr. Beides verbraucht nur unnützen Speicher, wenn Sie weder Erweiterungen noch die Uhr haben. Sie können diese Befehle ja wieder vom Orginal der Workbench auf die Arbeits-Diskette übertragen, wenn Sie sich eine Speichererweiterung mit Uhr zulegen o.ä.

Zum anderen gibt es einige speicherintensive Befehle wie AddBuffers oder Resident, die Speichermengen benötigen, die ohne Erweiterungs-Karte nicht unbedingt vorhanden sind. Auch RemRAD gehört dazu, obwohl das Kommando die reset-feste RAM-Disk entfernt, denn diese können Sie nur anlegen, wenn der Speicher dafür zur Verfügung steht.

Eine weitere Gruppe von CLI-Kommandos eignet sich nur für den Gebrauch bei intensiver Arbeit in dieser Bedieneroberfläche. Wenn Sie ihre Diskette von der Workbench aus formatieren, Verzeichnisse löschen oder erstellen und auch alle übrige Diskettenarbeit über die Workbench erledigen, wäre es wenig sinnvoll, eine solche Menge von CLI-Befehlen auf der Diskette zu speichern, die sowieso nicht gebraucht würden.

Anders liegt die Situation, wenn Sie das CLI fast täglich aufrufen und darüber einige Einstellungen oder Veränderungen vornehmen. Je nach ihren Ansprüchen sollten Sie überlegen, welche Funktionen jedes Kommando anbietet und ob diese von Ihnen unbedingt benötigt werden. Steht Ihnen der Speicher auf der Diskette zur Verfügung, sollte man jeden Befehl behalten. Wird er aber knapp, weil noch mehrere kleine oder ein großes Programm auf die Diskette gebracht werden müssen, geht es daran, abzuwägen.

Häufig wird das CLI, und das ist der dritte Anwendungskreis, auch zum Programmieren mit höheren Programmiersprachen wie C oder Pascal benötigt. Dann gibt es wieder neue Spezial-Kommandos und Befehle, die sich hier besonders anbieten. Auch das sollte man in Betracht ziehen.

Gerade aber auf diesem letzten Gebiet sollte man vielleicht überlegen, ob es nicht sinnvoll ist, eine separate Diskette für die Programmier-Arbeit anzulegen, auf der die gesamte Arbeitsumgebung darauf abgestimmt ist. So geht man keine faulen Kompromisse ein, bei denen bei jeder Situation etwas fehlt!

L:

Unser Lieblingsrechner, der Amiga, ist nicht nur für kurze Zeit konzipiert worden. Er wurde von Anfang an so aufgebaut, daß man jederzeit Ergänzungen einfügen kann. Aus diesem Grunde gibt es Teile im Betriebssystem, die fest eingebaut sind, weil sie einfach für den täglichen Betrieb notwendig sind, andere Teile wiederum werden eher selten gebraucht. Deswegen hat man sich gedacht, daß es hier am besten wäre, wenn die seltener gebrauchten Programme des Betriebssystems keinen festen Speicher wegnähmen und erst dazugeladen würden, wenn man sie ausdrücklich aufruft.

Wir haben hier das erste Verzeichnis vorliegen, das dieses Konzept unterstützt. Im L:-Directory sind alle Handler abgelegt, die es für den Amiga gibt, die aber nicht unter allen Umständen gebraucht werden. Außerdem gibt es hier natürlich Handler, die gar nicht bekannt waren, als das Betriebssystem entworfen wurde. Es ist also auch möglich, jederzeit einen Handler zu ergänzen. Aber Sie wissen bestimmt noch nicht, was ein Handler überhaupt ist.

Ein Handler ist ein Programm, das zur Ein- und Ausgabe genutzt wird. Beim Amiga gibt es viele verschiedene Geräte, die alle mit den gleichen Befehlen

angesprochen werden können. So können wir auf das Gerät PRT: genauso Daten schreiben wie auch auf das SPEAK:-Device. Für beide lautet der CLI-Befehl:

```
Type >PRT: filename
```

bzw.

```
Type >SPEAK: filename
```

Im ersten Fall wird der Text, der in der Datei filename enthalten ist, auf dem Drucker ausgegeben. Beim Aufruf des zweiten Befehls spricht der Amiga den Text, d.h. er gibt ihn über seine Audio-Ausgänge an den Monitor oder an die Stereo-Anlage in "sprachähnlichen" Lauten weiter.

Sie können sich aber sicher vorstellen, daß diese beiden Ausgaben sehr unterschiedlich sind.

Deshalb gibt es die Handler, die gleiche Befehle auf den jeweiligen Bereich umformen, so daß Sie als Benutzer viel weniger zu tun hätten. Im L:-Verzeichnis finden wir nun die neuen oder nicht so wichtigen Handler.

Wenn Sie freien Speicherplatz auf der Diskette herstellen wollen, müssen Sie überlegen, welche der Handler von Ihnen und Ihren Programmen nie oder sehr, sehr selten gebraucht werden. Hier ist die aktuelle Liste aller Handler:

```
Aux-Handler
FastFileSystem
Pipe-Handler
Ram-Handler
Speak-Handler
Disk-Validator
Newcon-Handler
Port-Handler
Shell-Seg
```

An der folgenden Tabelle sollten Sie ablesen können, welche Aufgaben die einzelnen Handler haben. Danach können Sie entscheiden, was Sie löschen möchten und welche der Handler wichtig sind:

Handler	Aufgabe
Aux	Audio-Ausgabe
Disk-Validator	Bereinigen der Disketten-Organisation
FastFileSystem	Schnellerer File-Zugriff mit neuem Format
NewCon	Neues Eingabe-Modul zur CLI-Editierung
Pipe	Datenablage
Port	Ein- und Ausgabeports
RAM	für die RAM-Disk
Shell	für das neue CLI-Shell
Speak	zur Sprachausgabe

Tabelle 2.1

S:

Das S:-Verzeichnis hat deswegen eine so wichtige Bedeutung, weil es gleich nach dem Start aufgerufen wird. In ihm sucht Kickstart, wenn es die Workbench bootet, nach der Startup-Sequence. Deswegen sollte man das Directory nicht löschen. Es ist aber wichtig zu wissen, daß auch Batch-Dateien hier hineingelegt werden können. Der Execute-Befehle sucht in diesem Verzeichnis nämlich selbständig, ohne daß man einen Pfad angeben muß. Sie sparen sich dadurch viel Tipp-Arbeit!

T:

Der Buchstabe T steht für engl. temporary und bedeutet zwischenzeitlich. Das soll heißen, daß in dieses Verzeichnis von jedem Programm Dateien abgelegt werden können, die nur für kurze Zeit gespeichert werden müssen. Viele Programme legen hier ihre Backups ab, die dann beim Anfertigen des nächsten gelöscht werden. Auch Ed macht dies! Sie können also hier nach der Version Ihres Textes vor der Veränderung suchen. Allerdings nehmen die abgespeicherten Backups und andere Files einigen Speicherplatz der Diskette weg. Man sollte hier deshalb von Zeit zu Zeit mit dem Befehl Delete aufräumen. Es ist nicht zu empfehlen, das Verzeichnis zu löschen, da viele Programme dieses ansprechen.

LIBS:

Hier sind alle Libraries abgelegt, die nicht im Kickstart untergebracht wurden, weil dort kein Platz war und man sie nur sehr selten benötigt. Sie können die Files aus dem Verzeichnis löschen, bei denen Sie meinen, daß Sie auf diese Library nicht zugreifen werden oder es die Programme garantiert nicht tun werden, die sich auf dieser Diskette befinden. Allerdings bricht jedes andere Programm ab, wenn es eine für sich wichtige Library dort nicht findet.

DEVS:

Das DEVS:-Verzeichnis teilt sich in drei große Bereiche, die für die Arbeit mit den Devices benötigt werden. Das erste Unterverzeichnis sind die Keymaps. In ihm sind alle Tastaturtabellen, die vom Keyboard.Device benutzt werden können, abgelegt. Auch die deutsche Tastaturtabelle findet man hier.

Auf neueren Workbench-Versionen ist nur noch diese Tastaturtabelle untergebracht, weil die anderen Speicherplatz verbrauchten, der nicht vorhanden war. Deshalb wurde auf der Extras-Diskette ein weiteres Keymaps-Verzeichnis angelegt, in dem die übrigen Tabellen abgelegt wurden. Das könnte wichtig sein, wenn Sie eine ausländische Tastaturbelegung erzeugen möchten.

Das zweite wichtige Unterverzeichnis aus dem DEVS-Ordner ist mit dem Namen Printer versehen. In ihm sind wiederum Datenfiles für ein weiteres Device, das Printer.Device, abgelegt. Dort sind alle Drucker-Treiber zu finden, die vom Amiga benutzt werden können. Auch hier gilt wieder das, was wir für die Tastaturtabellen gesagt haben. Wegen der allmählichen Überfüllung der Workbench-Diskette wurden die seltener gebrauchten Drucker-Treiber ausgelagert und auf die Extras-Diskette kopiert.

Mit dem letzten Unterverzeichnis ist ein externer Datenspeicher für das Clipboard.Device angelegt. Es sichert und überträgt innerhalb von Programmen Daten. Diese können auch zur Sicherung abgespeichert werden, wenn ein Programm die Daten nur an das Device sendet, sie aber nicht mehr abruft. Dafür ist dieses Verzeichnis vorgesehen.

Neben den drei Unterverzeichnissen sind aber noch andere Daten im DEVS-Ordner abgelegt. Und zwar finden wir hier, ähnlich wie bei LIBS:, die Devi-

ces vor, die erst später nachgeladen werden, weil ihre Wichtigkeit weniger hoch eingeschätzt wurde. Außerdem können so auch noch jederzeit externe Devices erzeugt werden, die nachgeladen werden müssen.

Als letztes sind im DEVS-Verzeichnis zwei Daten-Files zu finden. Einmal die MountList, in der neue Geräte für das System konfiguriert werden, und zum anderen die System-Konfiguration, in der alle wichtigen Daten zur System-Einrichtung gespeichert sind. Diese wird immer vom Programm Preferences beschrieben, wenn Sie einige Werte mit dem Programm verändern.

FONTS:

Wie Sie sicher wissen, ist es beim Amiga möglich, die Texte nicht nur in einem festgelegten Zeichensatz auszugeben. Zwar beschränkt sich die Workbench auf die Topaz-Fonts 8 und 9, aber andere Programme können jeden beliebigen Zeichensatz verwenden. Dafür gibt es im FONTS:-Verzeichnis eine Auswahl. Hier sind zusätzliche Fonts abgespeichert, die bei Bedarf geladen werden können.

Ein Möglichkeit, auf der Workbench-Diskette Platz zu schaffen, ist es, einige oder alle der hier abgespeicherten Zeichensätze zu löschen. Es ist dabei aber zu überlegen, ob man diesen Zeichensatz nicht benötigt. Denn manche Programme verlangen einfach, daß im FONTS:-Verzeichnis ein bestimmter Zeichensatz liegt. Ist das nicht der Fall, bricht das Programm einfach ab. Sie könnten dadurch also Fehlfunktionen haben, die sich nicht ergründen lassen. Denn es ist häufig schwer herauszufinden, welcher Zeichensatz fehlt oder ob der Fehler überhaupt daran lag.

Möchten Sie aber einen Zeichensatz oder gleich mehrere entfernen, geht man wie folgt vor:

Zuerst listen Sie sich das FONTS:-Verzeichnis auf.:

```
ruby (dir)
  12                                    15
  8
opal (dir)
  12                                    9
sapphire (dir)
  14                                    19
```

```
   diamond (dir)
     12                                20
   garnet (dir)
     16                                9
   emerald (dir)
     17                                20
   topaz (dir)
     11
diamond.font                         emerald.font
garnet.font                          opal.font
ruby.font                            sapphire.font
topaz.font
```

Sie finden mehrere Verzeichnisse vor sich, die den jeweiligen Namen des Zeichensatzes tragen. Außerdem gibt es sogenannte .font-Dateien, in denen Informationen zu den Fonts liegen. In den Font-Directories finden wir wiederum Dateien, die mit dem Namen der Punkt-Größe des Fonts belegt sind. Hier wählen Sie aus, welche Größen Sie benötigen und welche nicht. Ist das Verzeichnis ganz leer, können Sie auch die .font-Datei im FONTS:-Verzeichnis löschen. Solange aber auch nur eine Punkt-Größe bestehen bleibt, ist diese auf keinen Fall zu löschen.

Prefs

Das Programm Preferences, mit ihm werden alle wichtigen Einstellungen gemacht. Da es sich aber gezeigt hat, daß dieses Programm besonders bei speziellen Anwendungen viel häufiger als geplant aufgerufen wird, hat man sich gedacht, daß es dann doch lobenswert wäre, wenn man direkt in einen Teil des Programms einsteigen kann.

Das ist jetzt dadurch gelungen, daß es in diesem Verzeichnis neben dem Programm noch drei weitere Icons gibt, die nichts anderes machen, als das Programm aufzurufen und ihm die Informationen zu übergeben, in welchen Programmteil man einsteigen will. Da es sich nur um Icons, nicht aber um Programmteile handelt, bringt das Löschen sehr wenig zusätzlichen Speicher. Gleichermaßen verhält es sich mit dem "Programm" CopyPrefs. Es ist eine Batch-Datei, die über XIcon von der Workbench gestartet werden kann, und enthält nur zwei Befehle. Auch hier läßt sich so gut wie kein Speicher gewinnen, wenn man sie löscht.

Sie können bei diesem Verzeichnis nur überlegen, ob Sie nicht auf das gesamte Programm Preferences und alle seine Anhängsel verzichten wollen. Dann alleine lohnt sich eine Unternehmung dieser Art.

System

Neben den CLI-Kommandos gibt es noch die System-Programme oder -Befehle. Sie erfüllen gleichermaßen elementare Aufgaben, sind aber nicht unbedingt als Befehl aufzufassen. Man hat sie deshalb in diesem Verzeichnis untergebracht. Wir finden hier Format, DiskCopy und andere. Durch das Path-Kommando sucht das CLI oder die Shell auch hier nach einem eingegebenen Befehl. Allerdings sind nicht alle hier untergebrachten Programme unbedingt nützlich oder brauchbar. Sehen Sie deshalb hier zuerst eine Liste der ab Version 1.3 vorhandenen Programme:

```
CLI
DiskCopy
FastMemFirst
FixFonts
Format
InitPrinter
MergeMem
NoFastMem
SetMap
```

Das CLI-Programm ist in dieses Verzeichnis gerutscht, weil es im Hauptverzeichnis durch die Shell ersetzt wurde. Benötigen Sie das CLI gar nicht mehr, weil Sie nur noch mit der Shell arbeiten, könnte man denken, daß dieses Programm unnütz geworden wäre. Das ist aber nicht der Fall! Die Shell besteht nämlich nur aus einem Icon, das unser altes CLI-Programm auf eine etwas andere Art aufruft! Anstatt des alten Eingabe-Devices wird das NewCon-Device verwendet, mehr nicht. Also können wir dies auf keinen Fall löschen.

DiskCopy und Format werden von der Workbench nachgeladen, wenn man Duplicate oder Initialize für Disketten aufruft. Da diese beiden Funktionen sehr oft gebraucht werden, dürfen sie nicht entfernt werden. Gleichermaßen wichtig ist auch SetMap. Denn ohne dieses Kommando wäre die deutsche Tastaturbelegung nicht möglich.

FastMemFirst und NoFastMem sind zwei Speicher-Utilities, die Besitzern von Speicher-Erweiterungen Hilfestellung leisten. Das erste Programm sorgt dafür, daß der etwas schnellere Speicher zuerst belegt wird. Mit dem zweiten können wir Programme läuffähig machen, die zwar nur Chip-RAM gebrauchen können, deren Routinen dies aber nicht genau angeben. Durch Abschalten des Fast-RAM begegnen wir diesem Fehler. MergeMem ist auch ein neues Programm, das Speicherkarten, die hintereinander liegen, zusammenbindet und konfiguriert. Seine Funktion konnte von uns allerdings noch nicht getestet werden.

InitPrinter ist ein Programm, das den eingestellten Printer zurücksetzt und neu initialisiert. Mehr nicht! Wer keinen Drucker hat, benötigt es sowieso nicht, wer einen hat, kann auch darauf verzichten, weil ich persönlich es noch nie gebraucht habe!

Neu hinzugekommen ist das Utility FixFonts, das im Fonts-Verzeichnis aufräumt und alles neu organisiert, wenn man neue Fonts hineingelegt oder alte herausgenommen hat. Dadurch wird der Zugriff auf die Disketten-Fonts schneller. Es wird aber so selten gebraucht, daß man es ruhig löschen kann und bei Bedarf von dem Orginal der Workbench lädt.

Utilities

Während im System-Verzeichnis (s.o.) Programme untergebracht sind, die größtenteils für die Workbench oder das CLI unentbehrlich sind, haben wir es hier eher mit Programmen zu tun, die dem Benutzer bei seinen alltäglichen Arbeiten helfen sollen. Ein Format-Kommando, wie es im System-Directory liegt, ist von so elementarer Bedeutung, daß es nicht fehlen darf, die Besetzung des Utilities-Directories hingegen hängt weitgehend davon ab, welche Bedürfnisse Sie haben. Hier sehen Sie zuerst den Inhalt des Verzeichnisses:

```
Calculator
ClockPtr
CMD
GraphicDump
InstallPrinter
More
Notepad
PrintFiles
Say
```

Alle diese Programme sollen Ihnen das Leben mit dem Amiga erleichtern. Wenn Sie also Platz sparen wollen, müssen Sie entweder auf Komfort verzichten, oder Sie erwischen ein Programm, das Sie sowieso nicht benötigen. Mal sehen ...

Der Calculator und das NotePad sind hinreichend bekannt. Von meiner Seite aus möchte ich empfehlen, zumindest den Calculator zu löschen, weil er seine Arbeit mehr schlecht als recht vollbringt. Was er kann, können Sie auch im Kopf! Anders ist das beim NotePad. Zum Editor des CLI ist das eine wirkliche Alternative, besonders in Hinblick auf die verschiedenen Zeichensätze und Schriftattribute.

ClockPtr ist ein witziges und zugleich nützliches Utility, das mit Hilfe des Mauspfeils die aktuelle Uhrzeit anzeigt. Und zwar wird immer, wenn man auf der Workbench arbeitet, anstelle der Grafik für den Pfeil einfach die Uhrzeit eingeblendet. Diese Methode funktioniert nur nicht, wenn man mit anderen Programmen arbeitet. Auch eher eine Spielerei ist das Programm Say, mit dem englische Texte in phonetische Sprache umgesetzt werden und dann über die Audio-Ausgänge des Amiga ausgegeben werden. Sie können so den Amiga laut und deutlich - aber nur in Englisch - sprechen lassen. Für die Sprachausgabe in Programmen wird es nicht gebraucht und nimmt mehr Platz weg, als daß es wirklich nützt.

Mit GraphicDump, InstallPrinter und PrintFiles haben wir drei Drucker-Utilities, die Sie natürlich nur gebrauchen können, wenn Sie einen Drucker besitzen. Das Programm InstallPrinter gibt es noch nicht so lange. Es erlaubt ihnen, einen beliebigen Druckertreiber auf der Workbench zu installieren, ohne daß Sie dabei viel Arbeit haben. Dieses Programm benötigen Sie nur einmal, wenn Sie ihren Druckertreiber eingestellt haben, dann nicht mehr.

PrintFiles gibt Textdateien schnell auf dem Drucker aus. Das macht sich von der Workbench ganz gut, ist aber über das CLI viel schneller mit Type zu erledigen. Auf diesen Service können Sie ohne Bedenken verzichten, wenn Sie sich etwas im CLI auskennen. GraphicDump sollte man aber auf jeden Fall behalten. Mit einem kleinen Klick läßt sich der Inhalt der im Vordergrund liegenden Screen auf dem Drucker ausgeben. So einfach bekommen Sie das nicht noch einmal!

Um sich Text-Files auf dem Bildschirm schnell anzusehen oder durchzulesen, ist More unentbehrlich. Sie bekommen ohne dieses Utility nie wieder so

schnell Einblick in irgendwelche Dateien. Außerdem verbraucht es wenig Speicherplatz. Sind Sie allerdings nicht daran interessiert, Texte oder anderes von der Workbench aus zu betrachten, dann ist auch dieses Programm unnütz.

Sehr interessant und äußerst wirkungsvoll ist CMD. Es erlaubt das Abfangen von Nachrichten, die an den parallelen oder seriellen Port geschickt werden. So können Sie z.B. den Ausdruck abfangen, weil momentan kein Papier vorhanden ist und ihn später, weil er als File auf Diskette vorhanden ist, wieder an das entsprechende Gerät schicken.

Expansion

Das Expansion-Verzeichnis werden Sie fast immer leer vorfinden. In ihm sind keine Daten, die wir mit der Workbench mitgeliefert bekommen. Hier hinein müssen von uns Daten bzw. Programme gelegt werden, wenn es unser Amiga erfordert.

Man hat sich gedacht, daß nicht alle Hardware-Erweiterungen, sogenannte Expansionen, sofort vom Computer erkannt werden. Manche brauchen bestimmte Hilfsprogramme, die sie umstandslos in das Betriebssystem einbinden. Diese Erweiterungs-Hilfen sollen in diesem Verzeichnis liegen und werden aufgerufen, wenn der CLI-Befehl BindDrivers gegeben wird. Haben Sie keine Hardware-Erweiterungen, die diesen Weg gehen, kann das Verzeichnis getrost gelöscht werden.

2.2.2 Zahlensysteme

Wer erfolgreich mit dem Computer programmieren will, muß sich mit zwei neuen Zahlensystemen anfreunden: dem Dual- oder Binärsystem auf der Basis 2 und dem Hexadezimalsystem auf der Basis 16. Beide Zahlenformate können auch bei anderen Anwendungen nützlich sein. So kommt man z.B. nicht ohne das Hexadezimalsystem aus, wenn es darum geht, ein Programm mittels eines Speicher-/Disketten-Monitors (Programm zum Anzeigen von Speicherinhalten) zu untersuchen.

Dezimalsystem

Doch was versteht man eigentlich unter Zahlensystemen? Und wofür braucht man sie? Um diese Fragen zu beantworten, sehen wir uns zunächst das Dezimalsystem auf der Basis 10 an, mit dem wir normalerweise rechnen. Das Entscheidende an den uns vertrauten Systemen ist eine von rechts nach links steigende Wertigkeit. So ist eine "1" zehnmal mehr wert, wenn sie eine Stelle weiter links steht. Man sagt daher, daß das gebräuchliche Zahlensystem die Basis 10 hat. Auch wenn es uns keine Mühe macht, eine normale Zahl zu lesen, stellen wir uns jetzt erst einmal ganz dumm, und zerlegen die Zahl 1054 in ihre einzelnen Bestandteile.

```
  1054
= 1*1000 + 0*100  + 5*10   + 4*1
= 1*10^3 + 0*10^2 + 5*10^1 + 4*10^0
```

Hinweis zur Schreibweise: 10^3 soll 10 hoch 3 heißen (Potenzfunktion). 10^0 ist laut Definition und Logik gleich 1.

Wenn wir nun die einzelnen Teilwerte addieren, erhalten wir wieder unsere Zahl:

```
1*10^3 --> 1000
0*10^2 -->    0
5*10^1 -->   50
4*10^0 -->    4
           ====
Summe      1054
```

Dualsystem

Wir haben uns also im Kreis gedreht, da wir die Zahl zunächst zerlegt und dann wieder zusammengesetzt haben. Sinnvoll wird diese Technik, wenn wir mit dem Dualsystem arbeiten, das auf der Basis 2 aufbaut und daher nur die Ziffern 0 und 1 zuläßt. Jede Stelle ist bei diesem System doppelt soviel wert wie ihr rechter Nachbar. Es ist üblich, Dual- bzw. Binärzahlen immer das Prozent-Zeichen voranstellen, um sie von Dezimalzahlen zu unterscheiden. Dies ist übrigens keine Willkür, sondern eine bei Computern gebräuchliche Norm. Auch der Assembler (Hilfsmittel zur Erstellung von schnellen Programmen in Maschinensprache) arbeitet mit dem Prozent-Zeichen vor Binärzahlen. Doch

kommen wir zurück zu unseren Umrechnungen. Als binäre Beispielzahl wählen wir in diesem Fall %1100101:

```
  %1100101
= 1*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2
        + 0*2^1 + 1*2^0
```

Nun addieren wir wieder:

```
1*2^6 = 1*64 =  64
1*2^5 = 1*32 =  32
0*2^4 = 0*16 =   0
0*2^3 = 0* 8 =   0
1*2^2 = 1* 4 =   4
0*2^1 = 0* 2 =   0
1*2^0 = 1* 1 =   1
                ===
                101
```

Wir haben also die Binärzahl %1100101 in ein anderes Zahlensystem, und zwar das Dezimalsystem, umgewandelt. Als Ergebnis erhielten wird die Zahl 101.

Hexadezimalsystem

Doch wenden wir uns zunächst dem Hexadezimalsystem zu. Da es mit der Basis 16 arbeitet, müssen wir nicht mit 2 oder 10, sondern mit 16 unterschiedlichen Ziffern arbeiten. Leider haben wir jedoch nur 10 einstellige Zahlen zur Verfügung. Wir sind daher gezwungen, noch einige Buchstaben hinzuzuziehen.

Basis des Zahlensystems	Verwendete Ziffern
2	0,1
10	0,1,2,3,4,5,6,7,8,9
16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Damit wir normale und hexadezimale Zahlen unterscheiden können, werden wir ein Dollar-Zeichen vor alle Hex-Zahlen stellen. Dieses Zeichen wird auch

vom Assembler verwendet. C-Compiler tanzen jedoch aus der Reihe: Sie verlangen, daß Hexadezimal-Zahlen "0x" vorangestellt wird. Zur Übung rechnen wir die Zahl $FCE2 in das Dezimalsystem um:

```
  $FCE2
= $F*16^3 + $C*16^2 + $E*16^1 + $2*16^0
= 15*16^3 + 12*16^2 + 14*16^1 +  2*16^0
```

Der Übergang von Zeile 2 zu Zeile 3 der obenstehenden Rechnung bedarf noch einer genaueren Erklärung. Wir müssen nämlich noch darauf hinweisen, wie man die Buchstaben des 16er-Systems in normale Zahlen umwandelt. Dabei hilft uns und Ihnen die folgende Umrechnungstabelle:

Dezimal	Hexadezimal	Binär
0	$0	%0000
1	$1	%0001
2	$2	%0010
3	$3	%0011
4	$4	%0100
5	$5	%0101
6	$6	%0110
7	$7	%0111
8	$8	%1000
9	$9	%1001
10	$A	%1010
11	$B	%1011
12	$C	%1100
13	$D	%1101
14	$E	%1110
15	$F	%1111

Doch zurück zu unserer Hexadezimal-Zahl. Da wir nun den letzten Schritt verstanden haben, können wir wieder addieren.

```
$F*16^3 = 15*16^3 = 15*4096 = 61440
$C*16^2 = 12*16^2 = 12* 256 =  3072
$E*16^1 = 14*16^1 = 14*  16 =   224
$2*16^0 =  2*16^0 =  2*   1 =     2
                             =====
                             64738
```

Nun können wir vom 2er- und 16er- ins 10er-System umrechnen. Oft ist es auch wichtig, zwischen Binär- und Hexadezimal-Zahlen umzurechnen. Das ist sogar viel leichter. Beginnen wir mit der Umwandlung von Hex nach Bin. Da eine Stelle des Hexadezimal-Systems vier Stellen des Binär-Systems entspricht (16^1=2^4), können wir jede Stelle einzeln bearbeiten. Dabei ziehen wir unsere Umrechnungstabelle zu Rate. Dazu ein kleines Beispiel: Wir werden versuchen, $FCE2 als Dualzahl zu schreiben.

```
$FCE2

$       F    C    E    2
%    1111 1100 1110 0010

%1111110011100010
```

Und jetzt das gleiche rückwärts. Eigentlich müßten wir die Zeilen der obigen Rechnung nur in umgekehrter Reihenfolge aufschreiben. Da wir jedoch auch etwas verstehen wollen, gehen wir etwas ausführlicher vor.

Wir zerteilen die Zahl zunächst in Blöcke von je vier Ziffern. Sollte sich die Anzahl der Stellen nicht durch vier teilen lassen, schreiben wir so viele Nullen vor(!) die Zahl, bis es klappt. (%11001 - > %00011001). Diese vierstelligen binären Zahlen wandeln wir nun von links nach rechts mit unserer Umrechnungstabelle um.

```
%1111110011100010

%1111 --> $F
%1100 --> $C
%1110 --> $E
%0010 --> $2

$FCE2
```

Hilfsprogramm zur Umrechnung der Zahlensysteme

Rechenfaule können mit dem folgenden AmigaBASIC-Programm Werte in den drei Systemen umrechnen. Zunächst muß die Zahlenbasis gewählt werden, auf der die Eingabe erfolgen soll. Dann ist die Zahl selbst einzugeben. Ist dies geschehen, rechnet das Programm die Zahl in alle anderen Systeme um.

Listing:

```
' ********************************
' *                              *
' *  Programm: Zahlensysteme.BAS *
' *                              *
' *  Sprache:  AmigaBASIC        *
' *                              *
' ********************************

Anfang:
PRINT "Wählen Sie das Zahlensystem, in dem Sie die Zahl"
PRINT "eingeben wollen."
PRINT
PRINT "1 Dezimal"
PRINT "2 Hexadezimal"
PRINT "3 Dual"
PRINT
PRINT "0 Programmende"
PRINT
PRINT ">";
i$=""
WHILE i$<"0" OR i$>"3"
   i$=INKEY$
WEND
PRINT i$
If i$="0" THEN
   CLS
   END
ELSEIF i$="1" THEN
   PRINT "Dez>";
   INPUT "",d&
   If d&<0 THEN
      CLS
      PRINT "Zahl muß >= null sein"
      PRINT
      GOTO Anfang
   END If
ELSEIF i$="2" THEN
   d&=0
   PRINT "$";
   INPUT "",i$
   If i$="" THEN i$="0"
   FOR i%=1 TO LEN(i$)
      k$=""
      j$=UCASE$(MID$(i$,i%,1))
      If j$>="0" AND j$<="9" THEN k$=j$
      If j$>="A" AND j$<="F" THEN k$=j$
```

```
      If k$="" THEN
         CLS
         PRINT "Eingabe-Fehler"
         PRINT
         GOTO Anfang
      END If
   d&=d&*16+VAL("&h"+k$)
   NEXT
Else
   d&=0
   PRINT "%";
   INPUT "",i$
   If i$="" THEN i$="0"
   FOR i%=1 TO LEN(i$)
      j$=MID$(i$,i%,1)
      If j$="0" OR j$="1" THEN
         d&=d&*2+VAL("&h"+j$)
      Else
         CLS
         PRINT "Eingabe-Fehler"
         PRINT
         GOTO Anfang
      END If
   NEXT
END If
CLS
d$=STR$(d&)
d$=RIGHT$(d$,LEN(d$)-1)
h$=HEX$(d&)
b$=""
WHILE d&>0
   h=d&
   If d&/2=INT(d&/2) THEN a$="0" Else a$="1"
   d&=INT(d&/2)
   b$=a$+b$
WEND
If b$="" THEN b$="0"
PRINT "Dezimal.......";d$
PRINT "Hexadezimal...$";h$
PRINT "Dual.........%";b$
PRINT
GOTO Anfang
```

"Alles schön und gut, doch was bringt mir das?" Leider ist es so, daß das Rechnen mit fremden Zahlensystemen einem Computer-Neuling oft schwerfällt. Auch kann man nicht leicht einsehen, warum man in einer Computersprache, die mit "normalen" Zahlen umgehen kann, plötzlich mit ungewohn-

ten und auf den ersten Blick unpraktischen Zahlensystemen rechnen soll. Ursache allen Übels ist die Technik, mit der Computer intern arbeiten: die Digitaltechnik. Sie diktiert, mit welchen Werten der Computer besonders gut umgehen kann. Es fällt ihm z.B. sehr leicht, mit zwei zu multiplizieren, während er sich bei einer für uns simplen Multiplikation mit zehn mächtig anstrengt, da er alle Zahlen mit einer Kette von zwei Zuständen darstellen muß. Das Hexadezimalsystem wurde übrigens nur eingeführt, um auf die langen Dualzahlen verzichten zu können. Aus ihm läßt sich eine Dualzahl auch schneller ableiten als aus dem Dezimalsystem.

Dies überzeugt jedoch immer noch nicht. Sehen wir uns daher an einem praktischen Fall den Vorteil der neuen Systeme an. Wen die Standard-Funktionen des AmigaBASIC nicht mehr reizen, der beschäftigt sich mit systemnaher Programmierung (Kapitel 5 & 6), bei der direkt mit dem Betriebssystem des Rechners gearbeitet wird. Das Betriebssystem benutzt z.B. sogenannte Flag-Variablen. In ihnen werden mehrere Ja/Nein-Zustände festgehalten. Gibt man eine solche Variable im Zehnersystem aus, ist mit der direkt sichtbaren Information nichts anzufangen. Nur bei Darstellung im Binärsystem kann man genau sehen, welches Flag gesetzt ist, also auf "Ja" steht (alle Stellen der Binär-Zahl, an denen eine eins steht).

Ungewöhnliche Rechenzeichen und Zahlen

Wer tagtäglich mit Programmiersprachen arbeitet, verschwendet an das Thema dieses Kapitels nicht einen Gedanken. Ein- und Aufsteiger können mit den neuen Zahlendarstellungsformen des Computers oft nichts anfangen, da sich dieser nicht an die gewohnten Regeln hält. Dies betrifft z.B. die Zahlenschreibweise und die Rechenzeichen.

Fließkommazahlen (Zahlen mit Nachkommastellen) werden nicht wie gewohnt mit einem Komma, sondern mit einem Punkt eingegeben. Also nicht 3,1415, sondern 3.1415.

Auch bei einigen Rechenzeichen weicht Kollege Computer von den normalen Schreibregeln ab. Für die Multiplikation verwendet er das Sternchen (*) und für Divisionen den normalen Schrägstrichstrich (/). A hoch B muß im allgemeinen in der Form A^B eingegeben werden, wobei zu beachten ist, daß das

"Dach" (^) in BASIC die Hoch-Funktionen, in anderen Sprachen jedoch andere Operationen vertritt.

Bis jetzt war es nicht schwer. Bei sehr großen und sehr kleinen Zahlen verwenden viele Programme (z.B. AmigaBASIC) Exponentialdarstellungen, die sich aus optischen Gründen von der normalen Schreibweise unterscheiden, da der Amiga hochgestellte Zahlen nicht darstellen kann. Damit man solche Zahlen lesen kann, muß man ihren Aufbau kennen, der allgemein formuliert so aussieht:

```
(+/-)(Wert)E(+/-)(Hochzahl)
```

Um die uns bekannte Zahlendarstellung zu erhalten, müssen wir den Wert mit "10 hoch der Zahl hinter dem E" multiplizieren, die sowohl positiv als auch negativ sein kann. Negative Potenzen bilden Zahlen kleiner 1 (also solche, bei denen erst einmal viele Nullen hinter dem Komma stehen, bevor die eigentliche Zahl beginnt). Doch sehen wir uns das Format an einigen Beispielen an:

1. Beispiel

```
  1.2345E+09              So schreibt AmigaBASIC die Zahl
= 1.2345 * 10^9           Umwandlung in normale Form
= 1.2345 * 1000000000     Ausrechnen der Potenz
= 1234500000              Ergebnis der Multiplikation
```

2. Beispiel

```
  -3.49012E+07
= -3.49012 * 10^7
= -3.59012 * 10000000
= -35901200
```

3. Beispiel

```
  5.432E-08
= 5.432 * 10^(-8)
= 5.432 * 0.00000001      Vorsicht: Häufige Fehlerquelle
= 0.00000005432
```

Verwendet man in AmigaBASIC übrigens doppeltgenaue Zahlen, wird anstelle des "E" ein "D" verwendet.

Sinn der E-Zahlen

Im normalen Alltagsumgang kommt man auch ganz gut ohne E-Zahlen aus. Die Stärke der E-Zahlen liegt in der Möglichkeit, sehr große und sehr kleine Zahlen platzsparend anzugeben. Bei der Zahl 1E+70 erkennt man sofort, daß eine Eins mit 70 Nullen gemeint ist. Würde diese Zahl ausgeschrieben, müßte man mühevoll alle Nullen einzeln zählen. E-Zahlen kommen in Anwendungsprogrammen nicht besonders häufig vor. Sie dienen vornehmlich der Lösung mathematischer Probleme mit dem Computer. Natürlich können Sie auch leicht zur Speicherung von Atomgewichten oder Sonnenmassen herangezogen werden.

Ungenauigkeiten

Die Tatsache, daß mit Hilfe der E-Zahlen sehr große Zahlen benutzt werden können, darf nicht darüber hinwegtäuschen, daß diese Zahlen nicht mit allen 30, 40 oder noch mehr Stellen gespeichert werden. Der Computer nimmt bei derart großen Zahlen an, daß relativ kleine Werte nicht beachtet werden müssen. Sie können auch gar nicht beachtet werden, wenn eine einigermaßen kompakte Datenspeicherung gewünscht wird. Addiert man also in AmigaBASIC ständig den Wert 1 zu einer Variablen X, die den Inhalt 1E+30 hat, wird diese nie den Wert 1E+31 erreichen, weil die 1 im Verhältnis zur 1E+30 so klein ist, daß sie erst gar nicht addiert wird.

2.3 Sinnvolles Arbeiten im CLI

Arbeitet man im CLI, ist es ganz wichtig zu wissen, wie man Laufwerke, Drucker und andere Geräte ansprechen kann. Dieses Wissen ist jedoch auch außerhalb des CLI von Nutzen. Zum Beispiel immer dann, wenn ein Filename eingegeben werden muß. "Das kann doch nicht so schwer sein", werden Sie jetzt vielleicht sagen. Doch was ist, wenn sich die gewünschte Datei nicht in dem Verzeichnis befindet, von dem aus das Programm gestartet wurde bzw. auf das dessen aktueller Pfad zeigt? Dann muß man wissen, wie man in das Verzeichnis gelangt - und das ist für Einsteiger nicht immer ganz leicht.

2.3.1 Geräte und Pfade

Geräte

Doch klären wir zunächst, was der Amiga unter dem Begriff "Gerät" versteht. Prinzipiell gibt es drei verschiedene Klassen. Zunächst wären die realen Geräte zu nennen, die sich auf Hardware beziehen. "Haben nicht alle Geräte etwas mit Hardware zu tun?" Nein, der Amiga betrachtet auch eine bestimmte Fensterart als Gerät. Ein Fenster zu öffnen, ist z.B. ohne zusätzliche Hardware möglich. Diese Geräte werden vom Amiga nur mittels Software verwaltet. Die noch verbleibende dritte Gerätegruppe, die virtuellen Geräte, ermöglichen es, bestimmte Dateien, auf die ein Programm oder das Betriebssystem zugreifen muß, an jedem beliebigen Ort abzulegen. Das Programm greift dann automatisch auf die richtige Diskette, das richtige Gerät oder das richtige Verzeichnis zu. Bleibt noch anzumerken, daß man den Namen einer Diskette auch als Gerätenamen interpretieren kann. Man kann auf diese Weise erreichen, daß der Amiga mittels eines Requesters mitteilt, auf welche Diskette er zugreifen will. Bevor wir uns mit den Funktionen der einzelnen Geräte im Detail beschäftigen, sollten wir uns erst einmal einen Überblick verschaffen. Ob das betreffende Gerät "gemounted" werden muß oder nicht, ist durch ein "x" gekennzeichnet. Will man ein solches Gerät benutzen, muß man dies dem Amiga zuerst über den Befehl Mount mitteilen.

Name	Mount	Funktion
DFx:		Laufwerk (x für 0-3)
RAM:		Simuliertes Laufwerk im RAM (RAM-Disk)
RAD:	x	Reset-feste RAM-Disk
DHx:	(x)	Amiga-Festplatte
JHx:	x	Festplatte der PC-Karte
NIL:		Datenschlucker
SER:		Serielle (RS232) Schnittstelle
AUX:	x	Wie SER: - jedoch tauglich für Terminal
PAR:		Paralelle Schnittstelle
PRT:		Drucker
CON:		Text-Fenster mit Zeileneditor
NewCON:	x	Wie CON: - jedoch verbessert
RAW:		Text-Fenster (Vorsicht!)
SPEAK:	x	Sprachausgabe
PIPE:	x	Datenkanal
*		Aktuelles CLI-Fenster

Beginnen wir mit den Speichermedien. Dies sind im allgemeinen Diskettenlaufwerke. Am Amiga können wir maximal vier (drei externe) Laufwerke betreiben. Sie werden über die Gerätebezeichnungen DF0:-DF3: angesprochen. Dabei ist DF0: das erste, DF1: das zweite, DF2: das dritte und DF3: das vierte Laufwerk.

Über RAM: können wir ein weiteres Gerät ansprechen, das sich zwar wie ein normales Laufwerk verhält, tatsächlich aber gar nicht vorhanden ist, sondern vom Amiga nur simuliert wird. Die Daten, die man in der RAM-Disk ablegt, verbrauchen zwar einen Teil des Hauptspeichers, werden aber dafür extrem schnell geladen. Einziges Problem bei der Sache ist, daß die RAM-Disk (RAM:) bei einem Reset bzw. einem Systemabsturz ihren Inhalt verliert. Man sollte sie daher nur zum Zwischenspeichern von Daten und als Speicher für Anwendungsprogramme und CLI-Befehle nutzen, die man während der Arbeit oft benötigt. Auf diese Weise erspart man sich Diskettenwechsel und Zeit.

Ist man stolzer Besitzer einer Festplatte (s. Hardware-Kapitel), spricht man diese über DH0: an. Auch hier gibt es wieder die Möglichkeit, mehrere Geräte anzusteuern. Jedes erhält dann eine fortlaufende Nummer.

Sollten Sie irgendwann auf ein Gerät mit der Bezeichnung JH0: stoßen, handelt es sich auch um eine Festplatte, die allerdings nur in Verbindung mit dem SideCar bzw. der PC-Karte verwendet werden kann.

Neben den Speichermedien gibt es noch andere Geräte, die zwar auch zur Datenein- und -ausgabe bestimmt sind, sich jedoch dadurch von den Laufwerken und Festplatten unterscheiden, daß sie nicht in der Lage sind, Programme aufzunehmen und (!) wieder bereitzustellen, falls dies gewünscht wird. Sie dienen vielmehr der Kommunikation mit linearen Geräten, z.B. Drucker, Modem usw.

Mit SER: können wir ein an der RS232-Schnittstelle angeschlossenes Peripherie-Gerät betreiben. PAR: ist für den parallelen Port zuständig, über den bevorzugt Drucker angeschlossen werden. Man sollte sich allerdings davor hüten, SER: und PAR: direkt anzusprechen. Für den Drucker empfiehlt es sich, PRT: zu verwenden. Daten, die an PRT: geschickt werden, werden mit Hilfe der in Preferences eingestellten Werte (Druckertreiber, Papierbreite usw.) so umgewandelt, daß der Drucker sie richtig versteht.

Möchte man die serielle Schnittstelle (RS232) als Eingang verwenden, ist es ratsam, mit Workbench 1.3 zu arbeiten und statt SER: AUX: einzusetzen. In der Steuer-Software zu SER: gibt es nämlich einen kleinen Fehler.

Im CLI kann man das aktuelle Fenster auch als Ein- und Ausgabegerät verwenden. Dazu dient das Sternchen (*). Es wird im Gegensatz zu allen anderen Geräten ohne einen nachfolgenden Doppelpunkt eingegeben. Dieses Gerät ist jedoch mit einer gewissen Vorsicht zu genießen. Es ist z.B. möglich, eine ASCII-Datei mit

```
Copy Datei *
```

im CLI-Fenster auszugeben. Dabei sollte man jedoch beachten, daß man die Ausgabe nicht mit Ctrl C abbrechen kann, wie dies bei

```
Type Datei
```

möglich wäre. Bei längeren Texten kann dies ziemlich problematisch werden.

Der Bildschirm kann auch noch über drei weitere Gerätenamen angesprochen werden. CON: ist sowohl zur Eingabe als auch zur Ausgabe von Texten geeignet. Bei der Eingabe kann man wie im CLI-Fenster in der aktuellen Zeile noch korrigieren, bevor man die endgültige Zeile mit Return übernimmt.

CON: bedarf einer genaueren Erklärung, da ein Fenster geöffnet wird, dessen Größe man selbst festlegen kann. Dabei gibt man die Koordinaten der oberen linken Ecke sowie die Breite und Höhe des zu öffnenden Fensters an. Am Beispiel des NewCLI-Befehls sieht das so aus:

```
NewCLI CON:10/20/400/100/Fenstername
```

Geben Sie diesen Befehlsaufruf ein, wird ein Fenster geöffnet, das 10 Bildpunkte vom linken und 20 vom oberen Rand der Workbench-Screen entfernt ist. Es ist 400 Punkte breit und 100 Punkte hoch. Einen Namen hat es auch: Fenstername. Möchte man ein Fenster ohne Namen öffnen, muß man den letzten Schrägstrich übrigens auch angeben. Hierzu wieder ein kleines Beispiel:

```
Type ReadMe.ASC TO CON:0/0/640/200/
```

Auf den ersten Blick hätte es keinen Unterschied gemacht, wenn wir statt dessen

```
Type ReadMe.ASC TO RAW:0/0/640/200/
```

geschrieben hätten. Die eigentlichen Unterschiede von RAW: werden erst bei der Eingabe ersichtlich. Im Gegensatz zu CON: werden nämlich alle gedrückten Tasten und nicht die kompletten Zeilen übermittelt. Auch kann man einem RAW:-Fenster nicht mitteilen, daß man die Eingabe abschließen möchte.

Ist das NewCON-Device gemounted, kann es alternativ für CON: verwendet werden. Es hat den Vorteil, daß Eingaben innerhalb einer Zeile besser editiert und vorhergehende Zeilen wieder aufgegriffen werden können. Die erweiterten Editiermöglichkeiten können Sie der Beschreibung der DOS-Shell (am Ende des CLI-Kapitels) entnehmen, die ebenfalls das NewCON-Device benutzt.

Das nächste Gerät, das wir Ihnen vorstellen wollen, ist ein regelrechter Datenschlucker. Die Rede ist von NIL:. Alles, was Sie nach NIL: schicken, wird in keiner Weise weiterverarbeitet. Das liegt nun aber nicht daran, daß man vergessen hätte, die Routinen für NIL: fertigzustellen. NIL: hat nämlich die Aufgabe, Daten abzufangen, die nicht angezeigt werden sollen. Auf diese Weise kann man z.B. unerwünschte Meldungen von CLI-Befehlen unterdrücken.

Das PIPE-Device ist für Datenbewegungen zwischen verschiedenen Tasks zuständig. SPEAK: ist für die Sprachausgabe auf DOS-Ebene zuständig. Wenn Sie ein Text-File über dieses Device (Gerät) ausgeben, lädt der Amiga einige System-Programme und gibt mit deren Hilfe den Inhalt des Text-Files in Form von menschlicher Sprache wieder, bei der man allerdings einen starken amerikanischen Akzent erkennen kann.

Verzeichnisse

Da Disketten und andere Speichermedien besonders wichtig sind, kann man sie noch unterteilen, um auf diese Weise Ordnung herzustellen, indem man gleichartige Dateien in einem Ordner zusammenfaßt. Diese Ordner, die wir ab jetzt Verzeichnisse oder Unterverzeichnisse nennen werden, dienen dem gleichen Zweck wie die Schubladen eines Schrankes: der Übersichtlichkeit. Stellt

man sich die Diskette in Form einer Schrankwand vor, würden die Verzeichnisse den Schubladen entsprechen. Aus diesem Grund werden Unterverzeichnisse auf Workbench-Ebene auch oft als Schubladen dargestellt. Übrigens: Verzeichnisse können sich auch innerhalb von anderen Verzeichnissen befinden. Hauptsächlich werden in Verzeichnissen jedoch Dateien und Programme abgelegt. Das oberste Inhaltsverzeichnis einer Diskette bezeichnen wir als Hauptverzeichnis.

Wenn man den Namen eines Verzeichnisses oder eines Files angeben muß, kann dies auf zwei Arten geschehen: relativ oder absolut. Beim absoluten Verfahren ist alles anzugeben: das Gerät und alle über den gewünschten File oder Verzeichnis liegenden Verzeichnisse. Das andere System kommt unter Umständen mit einem kürzeren Pfad aus. Ein Pfad ist der Name, der ein Verzeichnis beschreibt. Man kann einem Pfad also genau entnehmen, welche Diskette und welches Unterverzeichnis angesprochen werden soll. Doch zurück zu den relativen Pfaden. Sie beziehen sich auf das sogenannte aktuelle Verzeichnis eines Programms oder eines CLI-Tasks. Dieses aktuelle Verzeichnis gibt an, auf welches Verzeichnis zugegriffen wird, wenn ein Dateiname ohne zusätzliche Ortsangaben verwendet wird.

Das allgemeine Format einer absoluten Pfad-Angabe kann wie folgt aussehen:

```
Gerät:Verzeichnis/Filename
```

Zur obenstehenden Formatangabe ist wohl eine kleine Beschreibung notwendig. Vor dem Doppelpunkt steht der Name des Geräts oder der Diskette. Danach kann ein Verzeichnis-Name folgen. Um ihn von dem eventuell nachfolgenden File-Namen zu trennen, verwenden wir den Schrägstrich. Bei einer Verzeichnis-Struktur, bei der mehrere Verzeichnis-Ebenen ineinander verschachtelt sind, werden die Namen der Verzeichnisse, über die man zu dem gewünschten File gelangt, ebenfalls mit Schrägstrichen getrennt.

Nehmen wir einmal an, wir hätten eine Diskette, auf der sich einige Dateien befinden, die in verschiedenen Unterverzeichnissen abgelegt sind. Dabei sollen sich im Hauptverzeichnis zwei Unterverzeichnisse befinden, denen wir die Namen "Programme" und "Daten" gegeben haben. Im "Programme"-Directory sollen sich drei weitere Verzeichnisse befinden: "BASIC", "C" und "Assembler". Das Wort "Directory" hat einen englischen Urspung und bedeutet "Verzeichnis". Im "BASIC"-Directory soll sich eine Datei mit dem Namen "PRIM.BAS" befinden, im "C"-Verzeichnis soll "DISPLAY.C" zu finden

sein, und im "Assembler"-Directory sollen sich gleich zwei Dateien befinden: "DEMO.ASM" und "MON.ASM". In dem von uns angelegten "Daten"-Verzeichnis sollen ein Unterverzeichnis mit dem Namen "Texte" und eine Datei "BILD.IFF" abgelegt sein. Im "Texte"-Verzeichnis soll sich schließlich die Datei "BRIEF.TXT" befinden.

In der nachfolgenden Abbildung haben wir die Struktur unserer fiktiven Beispieldiskette grafisch dargestellt.

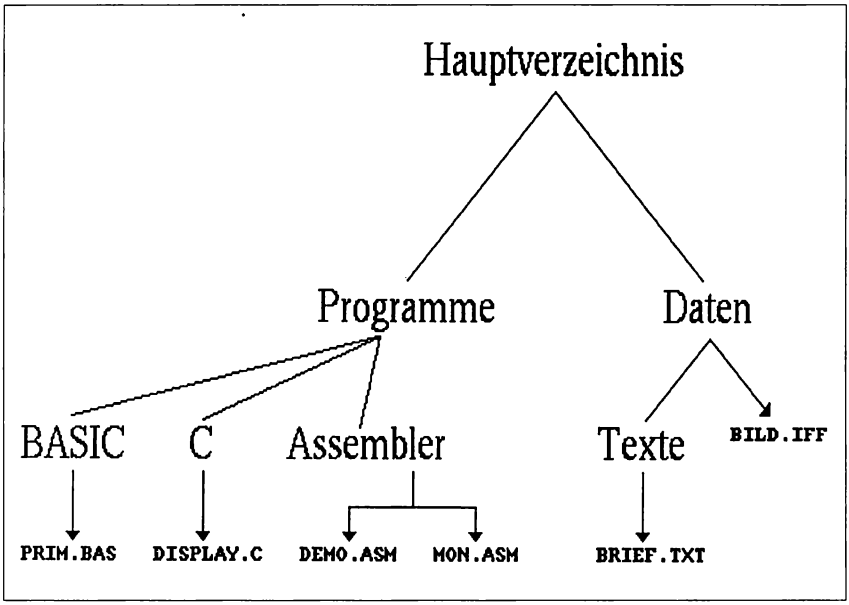


Abb. 2.3: Verzeichnisbaum

Fassen wir zusammen: Wir haben neben dem Hauptverzeichnis folgende Unterverzeichnisse:

```
Programme
Daten
BASIC
C
Assembler
Texte
```

Wenn wir weiterhin annehmen, daß die Beispieldiskette im internen Laufwerk eingelegt ist, können wir uns überlegen, wie wir diese Verzeichnisse bei Verwendung eines absoluten Pfades ansprechen müßten.

Für "Programme" und "Daten" ist dies ziemlich einfach. Wir müssen das Gerät (DF0:) und den Namen aneinanderhängen:

```
DF0:Programme
DF0:Daten
```

Jetzt wird es schon etwas schwieriger. Alle übrigen Directories befinden sich nicht im Hauptverzeichnis, das wir übrigens über DF0: ansprechen würden. Folglich müssen wir immer angeben, über welche höher gelegenen Verzeichnisse das gewünschte Directory erreicht werden kann (siehe grafische Darstellung).

```
DF0:Programme/BASIC
DF0:Programme/C
DF0:Programme/Assembler
DF0:Daten/Texte
```

Sicher können Sie sich jetzt schon denken, wie die Files anzusprechen sind. Sie werden genauso behandelt wie die Directories:

```
DF0:Programme/BASIC/PRIM.BAS
DF0:Programme/C/DISPLAY.C
DF0:Programme/Assembler/DEMO.ASM
DF0:Programme/Assembler/MON.ASM
DF0:Daten/BILD.IFF
DF0:Daten/Texte/BRIEF.TXT
```

Booten Sie bitte Ihre Workbench-Diskette, falls dies noch nicht geschehen ist, und öffnen Sie ein CLI-Fenster. Nun versuchen wir, die Verzeichnis-Struktur, die wir eben besprochen haben, in der RAM-Disk einzurichten. Dazu müssen Sie wissen, mit welchem Befehl ein neues Verzeichnis angelegt wird. Er heißt MakeDir. Zunächst legen wir "Programme" und "Daten" an. Dazu geben wir

```
MakeDir RAM:Programme
MakeDir RAM:Daten
```

ein. Wenn wir uns nun mit

```
Dir RAM:
```

das Inhaltsverzeichnis der RAM-Disk ausgeben lassen, erhalten wir folgende Ausgabe, der wir entnehmen können, daß unsere Anweisungen richtig ausgeführt wurden.

```
Daten (dir)
Programme (dir)
```

Die anderen Verzeichnisse können wir mit den folgenden Anweisungen erzeugen:

```
MakeDir RAM:Programme/BASIC
MakeDir RAM:Programme/C
MakeDir RAM:Programme/Assembler
MakeDir RAM:Daten/Texte
```

Zur Bestätigung rufen wir den Dir-Befehl mit der Option ALL auf, die bewirkt, daß auch alle Unterverzeichnisse aufgelistet werden.

```
Dir RAM: ALL
```

Der Aufruf liefert folgendes Ergebnis:

```
Daten (dir)
     Texte (dir)
Programme (dir)
     Assembler (dir)
     C (dir)
     BASIC (dir)
```

Um die Funktionsweise der relativen Pfade zu demonstrieren, löschen wir zunächst den kompletten Inhalt der RAM-Disk. Dazu benutzen wir den Ihnen sicher bekannten Delete-Befehl, dem wir allerdings das Schlüsselwort ALL übergeben, um auch den Inhalt aller Unterverzeichnisse zu löschen.

```
Delete RAM: ALL
```

Nun müssen wir uns mit einem neuen Befehl des CLI vertraut machen: CD. Diese Abkürzung steht stellvertretend für "Change Directory", was zu gut deutsch heißt, daß das aktuelle Verzeichnis eines CLI-Tasks geändert werden soll. CD kann aber außerdem noch zum Anzeigen des eingestellten Pfades

(Verzeichnisname) verwendet werden. Das aktuelle Verzeichnis erleichtert, wie wir eingangs bereits gesagt hatten, die Eingabe von Pfad-Namen. Ruft man CD ohne Parameter auf, zeigt der Amiga den Namen dieses aktuellen Directorys an. Übergibt man CD jedoch noch einen Pfad, wird dieser als akuelles Verzeichnis eingestellt. Probieren wir das doch gleich einmal aus, indem wir die beiden folgenden Anweisungen eingeben:

```
CD RAM:
CD
```

Mit dem ersten Aufruf des CD-Befehls haben wir RAM: zum aktuellen Verzeichnis erklärt. Der zweite Aufruf lieferte zur Bestätigung den aktuellen Pfad. Dieser lautet jedoch nicht RAM: sondern RAM DISK:. Hat der Amiga uns etwa nicht verstanden? Nein, im Gegenteil! Er denkt mit. Hätten wir für unser Beispiel nicht RAM:, sondern DF0: oder ein anderes Laufwerk gewählt, würde der aktuelle Pfad auch nicht DF0: heißen. AmigaDOS merkt sich nämlich immer den Namen der Diskette. Das hat den großen Vorteil, daß man, wenn man mit mehr Disketten arbeitet, als Laufwerke vorhanden sind, automatisch darüber informiert wird, welche dieser Disketten der Amiga gerade benötigt. Doch es gibt noch einen weiteren Vorteil dieser Technik. Hat man zum Beipiel nur ein Laufwerk, kann man mit

```
Dir DName:
```

das Inhaltsverzeichnis der Diskette ansehen, die den Namen DName trägt, da AmigaDOS automatisch über einen Requester verlangt, diese Diskette in ein beliebiges Laufwerk einzulegen. Benutzen Sie also mehrere Floppies, ist es Ihnen freigestellt, welches Laufwerk Sie benutzen wollen.

Doch zurück zum Thema. Da der aktuelle Pfad auf die RAM-Disk zeigt, können wir die ersten beiden Verzeichnisse anlegen, wobei wir allerdings nur den Namen angeben müssen, da der aktuelle Pfad schon vorgibt, daß RAM: benutzt wird.

```
MakeDir Programme
MakeDir Daten
```

Wir verzweigen mit

```
CD Programme
```

in das erste Verzeichnis. Das aktuelle Directory ist nun

```
RAM DISK:Programme
```

Sie können dies mit CD leicht nachprüfen. Nun können wir die Verzeichnisse anlegen, die in "Programme" untergebracht werden sollen:

```
MakeDir BASIC
MakeDir C
MakeDir Assembler
```

Jetzt müssen wir wieder ins Hauptverzeichnis verzweigen, um das "Daten"-Directory anzulegen. Dabei haben wir zwei Möglichkeiten:

```
CD /
```

und

```
CD :
```

Der erste Befehl bewirkt, daß in das nächst höhere Verzeichnis verzweigt wird. In unserem Fall ist das zufällig das Hauptverzeichnis. Die zweite Aufrufmöglichkeit führt direkt ins höchste Directory eines Geräts.

Nun legen wir noch die beiden Verzeichnisse "Daten" und "Texte" an. Zur Abwechselung werden wir, um "Texte" zu erzeugen, nicht nach "Daten" verzweigen. Und das geht so:

```
MakeDir Daten
MakeDir Daten/Texte
```

Da wir nun die Grundzüge der Verzeichnis-Anwendung kennen, können wir uns noch ein paar Kniffe ansehen. Man kann nämlich bei CD und anderen CLI-Befehlen, die mit Verzeichnissen und Pfaden arbeiten, auch sogenannte Querspünge verwenden. Um dies zu demonstrieren, geben wir

```
CD RAM:Programme/BASIC
```

ein. Um nun nach RAM:Daten/Texte zu kommen, haben wir zwei Möglichkeiten:

```
CD //Daten/Texte
```

oder

```
CD :Daten/Texte
```

Im ersten Beispiel springen wir mit den beiden Bruchstrichen zunächst je ein Verzeichnis nach oben. Von da aus gehen wir im Verzeichnis-Baum wieder nach unten. Das zweite Beispiel ist ein Zeichen kürzer. Mit dem Doppelpunkt wird zunächst das Hauptverzeichnis angewählt. Dann folgt der übliche Weg nach unten.

Schließlich wollen wir noch das File DF0:Prefs/Preferences nach RAM:Programme kopieren. Dazu bieten sich mehrere Lösungswege an:

```
Copy DF0:Prefs/Preferences RAM:Programme
Copy DF0:Prefs/Preferences RAM:Programme/Preferences
Copy FROM DF0:Prefs/Preferences TO
                                    "RAM DISK:Programme"
```

Nun sind die virtuellen Geräte an der Reihe. AmigaDOS gestattet es, eigene "Speichermedien" zu definieren, die über einen "Gerätenamen" angesprochen werden, obwohl ihre Daten sich tatsächlich in normalen Verzeichnissen auf anderen Geräten befinden. "Und wo liegt da der Vorteil?", werden Sie jetzt fragen. Anhand eines Beispiels wird dies deutlich: Alle CLI-Befehle, die, wie Sie sicher wissen, von Diskette nachgeladen werden, befinden sich in der Regel im "c"-Verzeichnis der System-Diskette. AmigaDOS spricht sie jedoch nicht über DF0:c oder Workbench-Diskette:c, sondern über das virtuelle Device C: an. Wenn Sie das Directory mit den CLI-Befehlen von der Workbench auslagern wollen, z.B. auf eine Festplatte, können Sie mit dem Assign-Befehl einstellen, daß C: nun nicht mehr auf Workbench-Diskette:c, sondern DH0:CLI-Befehle zeigen soll. AmigaDOS ist dies vollkommen egal. Es greift über C: auf seine Befehle zu und benutzt daher automatisch das von Ihnen eingestellte Verzeichnis. Um eine Liste aller bereits definierten virtuellen Devices zu erhalten, geben Sie bitte

```
Assign
```

ein. Daraufhin lieferte der Computer folgendes:

```
Volumes:
RAM DISK [Mounted]
Workbench1.3 [Mounted]
```

```
Directories:
CLIPS           RAM DISK:clipboards
ENV             RAM DISK:env
T               RAM DISK:t
S               Workbench1.3:s
L               Workbench1.3:l
C               Workbench1.3:c
FONTS           Workbench1.3:fonts
DEVS            Workbench1.3:devs
LIBS            Workbench1.3:libs
SYS             Workbench1.3:

Devices:
PIPE AUX SPEAK NEWCON DF0
PRT PAR SER RAW CON
RAM
```

Diesen Daten ist zu entnehmen, welche Disketten (Volumes) dem Amiga bekannt sind, und welche davon zur Zeit eingelegt (Mounted) sind. Die Aufzählung unter Directories gibt an, welche virtuellen Geräte definiert sind. Unter Devices finden wir schließlich noch eine Liste aller richtigen Geräte, die dem Amiga zur Zeit bekannt sind. Sollten Sie hier PIPE:, AUX: usw. vermisssen, kann dies daran liegen, daß Sie diese noch nicht eingebunden haben (z.B. Mount AUX:).

Nehmen wir einmal an, wir hätten eine Diskette, auf der sich sehr viele Zeichensätze befinden. Wenn diese Diskette den Namen ZS_Disk hätte, könnten wir das virtuelle Device FONTS: mit der Anweisung

```
Assign FONTS: ZS_Disk:
```

auf diese Diskette umlegen. Alle Programme (z.B. DPaint III), die nun das FONT:-Verzeichnis auslesen, greifen jetzt automatisch auf die neue Diskette zu. Um wieder das normale FONT-Directory auf der Workbench zu benutzen, können wir

```
Assign FONTS: SYS:fonts
```

eingeben. Von den virtuellen Devices können Sie auch in eigenen Programmen Gebrauch machen. Wenn Sie z.B. einen Vokabel-Trainer in BASIC geschrieben haben, können Sie mit dem BASIC-Befehl

```
CHDIR "VOK:"
```

veranlassen, daß nun alle Dateien aus dem VOK:-Directory geladen werden. Im CLI legen Sie dann einfach, bevor Sie das BASIC-Programm starten, fest, wo das VOK:-Verzeichnis liegen soll, ohne Änderungen an dem vielleicht sogar compilierten Programm vorzunehmen. Möchten Sie ein virtuelles Device auflösen, können Sie dies im vorgenannten Beispiel mit

```
Assign VOK:
```

tun. Und noch etwas sollten Sie wissen: Wenn in einem beliebigen Laufwerk eine Diskette eingelegt ist, die den gleichen Namen hat wie ein virtuelles Device, greift AmigaDOS auf diese Diskette zu, wenn das virtuelle Device angesprochen wird. Sie können diesen Sachverhalt natürlich ausnutzen! Nennen Sie z.B. Ihre Diskette, auf der sich Ihre Zeichensatz-Sammlung befindet, FONTS, und legen Sie diese ein. Bevor ein Programm auf FONTS: zugreift, werden automatisch die Zeichensätze von Ihrer Diskette verwendet, ohne daß Sie Assign benutzen müssen. Mit diesem Trick können Sie auch Platz auf der Workbench sparen, indem Sie zum Beipiel "fonts" komplett löschen. Wenn Sie nun von dieser Diskette booten, wird erst gar kein Eintrag FONTS: in der Liste der virtuellen Devices erzeugt, was zur Folge hat, daß Sie immer dann, wenn ein Programm Zeichensätze benötigt, aufgefordert werden, die Diskette mit dem Namen FONTS einzulegen!

2.3.2 Kleine und praktische Tips

ASCII-Dateien anlegen

Ist es nicht sehr ärgerlich, wenn eine ASCII-Datei angelegt werden soll und Ed nicht verfügbar ist? In diesem Fall besteht, jedenfalls, solange es sich nur um eine kleine Datei handelt und der Copy-Befehl greifbar, sprich ladbar, ist, kein Grund zur Panik. Man gibt den Dateinamen als Ziel und das aktuelle Fenster (vertreten durch ein *) als Quelle des Befehls an. Mit

```
Copy * S:StartUp-Sequence
```

kann man z.B. ein neues Start-File erstellen. Nachdem Sie diesen Befehl aufgerufen haben, wird alles, was Sie eintippen, in das angegebene File geschrieben. Korrekturen sind allerdings nur in der aktuellen Zeile möglich. Mit

Ctrl \ gibt man dem Copy-Befehl zu erkennen, daß keine weiteren Eingaben mehr folgen.

Anstelle von * hätten wir auch ein neues Fenster über CON: oder NewCON: öffnen können. Auch dann hätten wir unsere Eingabe mit Ctrl \ beenden müssen. RAW: dürfen wir jedoch auf keinen Fall verwenden, da es uns dann nicht möglich wäre, das Fenster wieder zu schließen, um zurück ins CLI zu gelangen.

Die Zeit aus der Echtzeituhr

Zählen Sie sich zu den glücklichen Besitzern einer Speicherkarte mit Echtzeituhr, kann es sein, daß Sie auch zu denjenigen gehören, die diese Uhr anzeigen, auslesen oder stellen wollten und dazu im Handbuch keine Informationen fanden? Uns ist dieser Mißstand wohl bewußt.

Beginnen wir mit dem Stellen der Uhr. Der Amiga betreibt unabhängig von der Echtzeituhr eine eigene System-Zeit, auf die von diversen Programmen zugegriffen wird. Sie kann mit dem CLI-Befehl Date gestellt werden. Hinter Date wird entweder das Datum, die Uhrzeit oder beides angegeben. Beispiele:

```
Date 13-Dec-88              --> 13 Dezember 1988
Date 17:30                  --> 17 Uhr 30
Date 13-Dec-88 17:30:00     --> 13 Dezember 1988;
                                17 Uhr 30
```

Das Datum muß mit Minus-Zeichen, die Uhrzeit mit Doppelpunkten eingegeben werden. Der Monat wird englisch abgekürzt: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.

Unser Ziel war es jedoch, die Echtzeituhr zu stellen. Dazu müssen wir noch

```
SetClock Save
```

eintippen, wodurch die Zeit aus der Amiga-Uhr in die Uhr der Karte übernommen wird. Will man diese Zeit auch verwenden, muß man in die StartUp-Sequence jeder Diskette, von der man den Amiga zum Arbeiten bootet, den Befehl

```
SetClock Load
```

einbauen, der die Zeit der Echtzeituhr an den Amiga übermittelt. Man muß SetClock dazu natürlich ins C-Verzeichnis der betreffenden Diskette kopieren. Dort sollte man auch Date ablegen, da Date von SetClock aufgerufen wird, um die Zeit anzuzeigen. Möchte man die Ausgabe allgemein unterdrücken, kann man das Gerät NIL: verwenden:

```
SetClock >NIL: Load
```

Hex-Dumps mit Type

Daß man reine ASCII-Texte mit dem Befehl Type anzeigen kann, ist hinreichend bekannt. Möchte man sich aber die Texte in einem normalen Programm ansehen, wird man feststellen müssen, daß es nicht ausreicht, diese einfach anzugeben, da sie neben normalen Zeichen noch solche Codes enthalten, die das CLI als Steuerzeichen auslegt. Das kann z.B. zur Folge haben, daß in den alternativen Zeichensatz umgeschaltet wird, der es unmöglicht macht, Texte zu entziffern. Um dieses Problem zu vermeiden, kann man Type anweisen, einen sogenannten Hex-Dump auszuwerfen, bei dem Sonderzeichen durch einen Punkt vertreten werden. Zusätzlich werden noch die Codes der Zeichen in hexadezimaler Schreibweise angezeigt. Wünschen wir einen Hex-Dump, stellen wir dies mit Opt H ein, wobei H für Hex steht. Beispiel:

```
Type SYS:Prefs/Preferences hex
```

Reine ASCII-Texte kann man übrigens auch mit Zeilennummern ausgeben. Dazu stellt Type eine weitere Option zur Verfügung: Opt N (Numerierung). Beispiel:

```
Type S:StartUp-Sequence number
```

Ein- und Ausgaben umleiten

CLI-Befehle erwarten Eingaben von der Tastatur und geben Daten in das CLI-Fenster aus. So war es jedenfalls bisher. Um das Verzeichnis einer Diskette, ein File, einen Hex-Dump oder etwas anderes auf den Drucker auszugeben, muß man die Ausgaben der CLI-Befehle auf den Drucker umleiten.

Für diesen Zweck stellt das CLI die Zeichen > und < zur Verfügung. > leitet die Ausgabe und < die Eingabe um. Man muß die Datenumleitung zwischen dem CLI-Befehl und dessen Parametern plazieren. Möchten Sie z.B. das komplette Verzeichnis der Diskette in DF1: auf den Drucker ausgeben, empfiehlt sich die folgende Anweisung:

```
Dir >PRT: DF1: all
```

NIL: ist bei der Datenumleitung von besonderem Interesse. Möchte man Meldungen unterdrücken, schickt man sie einfach nach NIL:

```
EndCLI >NIL:
```

Einige Befehle stellen auch ein eigenes Schlüsselwort bereit, um die Ausgabe auf ein bestimmtes Gerät umzuleiten. Bei Type geht es z.B. mit TO:

```
Type TextDatei TO PRT:
```

Files verschieben mit Rename

Mit dem CLI-Befehl Rename kann man Dateien nicht nur umbenennen, sondern auch verschieben. Möchte man ein File nur umbenennen, muß man bei beiden Parametern den gleichen Pfad angeben. Um eine Datei zu verschieben, wählt man beim ersten Parameter den Pfad, über den die Datei zu finden ist, und beim zweiten das Verzeichnis, in das das File verschoben werden soll. Beide Verzeichnisse müssen sich allerdings auf der gleichen Disketten befinden. Achten Sie darauf, kein Verzeichnis in ein tiefer gelegenes Unterverzeichnis zu verschieben (also nicht DF0:Dir1 nach DF0:Dir1/Dir2). Es könnte sonst aus dem Directory gelöscht werden. Zum Abschluß noch ein kleines Beispiel.

```
Rename SYS:System/Clock SYS:Uhr
Rename SYS:System/Clock.info SYS:Uhr.info
```

Das Programm Clock wird mit seinem .info-File ins Hauptverzeichnis verlegt. Dabei erhält es auch gleich einen neuen Namen. Man kann diese Operation zwar viel einfacher über die Workbench ausführen, sie ist natürlich nur als Beispiel interessant. Programme und Daten, die nicht mit einem Icon ausgestattet sind, müssen mit dieser Methode verlegt werden.

Verzögerte Befehlsausführung

Das CLI verfügt über eine Möglichkeit, ein Programm zu laden, ohne es sofort auszuführen. Erst wenn die [Return]-Taste betätigt wird, startet AmigaDOS das Programm. Dazu dient das Plus-Zeichen (+). Wird es mit einer Leerstelle Abstand hinter den Programm-Aufruf eingegeben, lädt das CLI zwar das Kommando, wartet jedoch darauf, daß Sie [Return] drücken.

```
Install DF0: +
```

In Verbindung mit dem Run-Befehl eröffnet das Plus-Zeichen weitere Möglichkeiten. Mit ihm ist es nämlich möglich, einen Hintergrundtask zu starten, der nicht nur einen, sondern mehrere Befehle ausführt, ohne dabei den umständlichen Weg über die Execute-Files zu gehen. Dies ist z.B. sinnvoll, wenn mehrere Texte gedruckt werden sollen:

```
Run Type Text1.ASC TO PRT: +
Type Text2.ASC TO PRT: +
Type Text3.ASC TO PRT:
```

Abbrechen eines Hintergrundtasks

Angenommen, wir wollen ein File ausdrucken und zugleich im CLI weiterarbeiten. Kein Problem werden Sie sagen, dafür gibt es doch den Type-Befehl. Und wenn man Run davor setzt, druckt der Amiga nebenbei:

```
Run Type >PRT: Dateiname
```

Doch wie können wir einen Befehl abbrechen, der im Hintergrund läuft? Hätten wir auf Run verzichtet, würde uns ^C ([Ctrl] [C]) weiterhelfen. Diese Tastenkombination versagt jedoch im vorliegenden Fall.

Aber auch für dieses Problem bietet das CLI eine Lösungsmöglichkeit in Form zweier Befehle: Status und BREAK. Geben wir zunächst Status ein. Die Ausgabe, die wir darauf erhalten, könnte etwa so aussehen:

```
Process  1: Loaded as command: status
Process  2: Loaded as command: type
```

Der Amiga teilt uns also mit, welche CLI gerade vorhanden sind und welche Anweisungen sie ausführen. Wie Sie sehen, taucht auch Type in dieser Liste auf. Intern macht es nämlich keinen Unterschied, ob ein Programm normal oder über Run gestartet wurde. In beiden Fällen wird ein CLI eingerichtet, das im Falle eines Run-Aufrufs jedoch über kein eigenes Fenster verfügt, und daher bei Bedarf seine Daten in dem Fenster ausgibt, von dem aus Run aufgerufen wurde.

Uns interessiert nun die Nummer des CLI (Process), in dem unsere Type-Anweisung läuft (hier ist es die zwei). Wenn wir uns diese Zahl gemerkt haben, rufen wir den BREAK-Befehl auf, dem wir als Parameter die Process-Nummer und das C übergeben:

```
BREAK 2 C
```

Wenn wir nun einige Sekunden warten und anschließend wieder Status aufrufen, werden wir feststellen, daß Type in der Liste nicht mehr auftaucht.

Übrigens: Wenn Sie eine Batch-Datei, die im Hintergrund arbeitet, abbrechen wollen, hilft Ihnen

```
BREAK <Nummer> D
```

weiter. Für <Nummer> ist dann wieder die Zahl einzusetzen, die man über Status erfährt.

Ein kleiner Trick mit dem Copy-Befehl

So gut der Copy-Befehl auch sein mag, er bietet doch eine kleine Tücke. Angenommen, das aktuelle Verzeichnis zeigt auf DH0:Testverzeichnis. Wenn Sie nun die Datei DF0:File in dieses Verzeichnis kopieren wollen, müssen Sie, da Copy immer einen Parameter verlangt, der angibt, wohin oder unter welchem Namen kopiert werden soll, unnötig viele Zeichen eintippen - und das kostet besonders bei langen File-Namen viel unnötige Zeit. Daher kann man anstelle von

```
Copy DF0:File File
```

auch

```
Copy DF0:File ""
```

schreiben. In beiden Fällen wird die angegebene Datei in das aktuelle Verzeichnis kopiert. In unserem Beispiel ist dies DH0:Testverzeichnis.

Bildschirmausgabe anhalten

Wenn Sie ein Text-File, das länger als eine Bildschirmseite ist, mittels Type in einem CLI-Fenster ausgeben, um es durchzulesen, können sie die Ausgabe eines Files in einem (New)CON:-Fenster (um ein solches handelt es sich bei einem CLI-Task) anhalten, indem Sie die Leerstaste betätigen. Wenn Sie nun [Backspace] drücken, wird die Ausgabe fortgesetzt.

Ist Ihnen dieses Verfahren zu umständlich, können Sie sich auch der rechten Maustaste bedienen. Ist das Fenster aktiv, wird die Ausgabe so lange unterbrochen, bis Sie die Taste wieder freigeben. Diese Technik eignet sich, da sie sehr einfach ist, besonders gut für unser eingangs geschildertes Problem.

2.3.3 Die Selbstdarstellung der CLI-Befehle

Passiert es Ihnen nicht auch oft, daß Ihnen die Syntax eines Befehls entfällt und Sie wohl oder übel zum Handbuch greifen müssen, um weiterarbeiten zu können? Wenn ja, dürfte Ihnen der folgende Trick sehr gelegen kommen. Alle CLI-Befehle zeigen bei Angabe des Fragezeichens nämlich den richigen Aufruf an. Doch nicht nur das! Sie fordern Sie sogar zur erneuten Eingabe auf, was Ihnen ein erneutes Laden des Befehls erspart. Dieser unscheinbare Vorteil hat es in sich. Wenn man nämlich nur ein Laufwerk besitzt, kann es leicht vorkommen, daß man einen Befehl nur aufrufen kann, wenn sich dieser in der RAM-Disk befindet. Das gilt z.B. für Install, den Befehl, mit dem es möglich wird, Disketten bootfähig zu machen. Geben Sie

```
Install ?
```

ein, teilt Ihnen der Befehl mit, was Sie eingeben sollen:

```
DRIVE/A,NOBOOT/S,CHECK/S:
```

Nun können Sie in aller Ruhe die Diskette einlegen, auf die Sie den Befehl anwenden möchten. Dann geben Sie noch

```
DF0:
```

ein - und schon ist Ihre Diskette bootfähig. Ohne das Fragezeichen wäre es ziemlich umständlich gewesen, dies zu erreichen. Hätten Sie nämlich einfach

```
Install DF0:
```

eingetippt, wäre die Diskette bearbeitet worden, von der aus der Install-Befehl geladen wurde.

2.3.4 Das Parameter-Format der CLI-Befehle

Wie wir bereits festgestellt hatten, kann man mittels des Fragezeichens, das man einem CLI-Befehl als Parameter übergibt, dessen Aufruf-Syntax ermitteln. Wenn man sich die Syntax genau anschaut, wird man feststellen, daß die Abkürzungen /A, /S und /K häufig hinter den Parametern zu finden sind. Was es mit ihnen auf sich hat, werden wir nun zuerst allgemein und dann an einigen Beispielen erläutern.

Einem CLI-Befehl kann man Informationen in zwei verschiedenen Formen übergeben: als Schlüsselworte und als Parameter. Ein Parameter ist variabel. Übergibt man z.B. einen File-Namen, ist dies ein Parameter. Schlüsselworte können nur bestimmte Worte sein. Je nach Befehl sind andere (oder auch gar keine) Schlüsselworte zugelassen. Sie sind dazu da, den nachfolgenden Parameter zu beschreiben. An einem Beispiel wird das deutlich: Der Copy-Befehl läßt unter anderem die Schlüsselworte FROM und TO zu. Dabei kann man FROM vor den Parameter, der die Quelle angibt, und TO vor den, der das Ziel angibt, plazieren. Gibt man beide Parameter (Quelle und Ziel) in der richtigen Reihenfolge an, kann man auf die Schlüsselworte ganz verzichten. Die folgenden Copy-Aufrufe sind daher gleichwertig:

```
Copy Quelldatei Zieldatei
Copy FROM Quelldatei Zieldatei
Copy Quelldatei TO Zieldatei
```

```
Copy FROM Quelldatei TO Zieldatei
Copy TO Zieldatei FROM Quelldatei
```

Notwendig sind die Schlüsselworte nur bei dem letzten Beispiel, da hier die Parameter in einer ungewöhnlichen Reihenfolge angegeben wurden.

Eine besondere Form der Schlüsselworte sind die Flags. Schlüsselwörtern mit Flag-Funktion folgt kein Parameter. Allein die Tatsache, daß das Flag angegeben bzw. gesetzt worden ist, bewirkt eine besondere Reaktion des Befehls.

Doch kommen wir zurück zu /A, /K und /S. Der nachfolgenden Tabelle können Sie die Bedeutung dieser Zeichen entnehmen:

Typ	Bedeutung
/A	Der zugehörige Parameter muß angegeben werden.
/K	Das Argument muß mit Schlüsselwort angegeben werden.
/S	Zum Schlüsselwort gehört kein Parameter (Flag).

/A und /K können kombiniert werden.

Der Copy-Befehl hat folgende Syntax:

```
FROM,TO/A,ALL/S,QUIET/S,BUF=BUFFER/K,CLONE/S,DATES/S,
     NOPRO/S,COM/S:
```

Die Parameter zu FROM und TO müssen angegeben werden (/A bezieht sich hier auf beide). ALL, QUIET, CLONE, DATES, NOPRO und COM sind Schlüsselworte. BUF oder BUFFER müssen zwar nicht angegeben werden, wenn jedoch der zugehörige Parameter verwendet werden soll, muß auch BUFFER eingegeben werden (/K). Alternativ kann man auch BUF verwenden. Das Gleichheitszeichen signalisiert, daß beide Schlüsselworte gleichbedeutend sind.

2.3.5 Dateinamen mit Sonderzeichen

Unter bestimmten Umständen kann es nötig sein, einen Dateinamen (oder allgemein einen Parameter) beim Aufruf eines CLI-Befehls in Anführungszeichen zu setzen. Dies ist z.B. dann notwendig, wenn der Parameter eine oder

mehrere Leerstellen enthält. Nun könnte man vermuten, daß man z.B. in Dateinamen, die man aus irgendeinem Grund als Parameter verwenden will, keine Anführungszeichen verwenden darf. Um dieses Problem zu umschiffen, wurde ein System geschaffen, das auch solche Fälle zuläßt. Dateinamen dürfen alle Zeichen außer dem Doppelpunkt (:) und dem Schrägstrich (/), der auch als Slash bezeichnet wird, enthalten. Nehmen wir an, wir wollen den Dateinamen

```
* = "A"
```

verwenden. Dazu müssen wir den Namen zuerst in Anführungszeichen setzen, da in ihm unter anderem zwei Leerzeichen vorkommen. Zu unserem großen Entsetzen kommen jedoch auch im Namen selbst Anführungszeichen vor. Um AmigaDOS zu verdeutlichen, daß es sich bei diesen Anführungszeichen nicht um die Begrenzung eines Parameters handelt, müssen wir vor jedes im Dateinamen vorkommende Anführungszeichen ein Sternchen (*) setzen. Doch was ist mit dem Sternchen am Anfang des Dateinamens? Ganz einfach: Um ein Sternchen im Dateinamen zu erhalten, muß man zwei Sternchen schreiben.

```
"** = *"A*""
```

Mit dem oben genannten System können wir Dateien und Programme mit sehr ausgefallenen Dateinamen nicht nur bearbeiten, sondern auch starten.

Wer glaubt, daß man, wenn man keine Sonderzeichen in Filenamen verwendet, auch keinen Ärger mit ihnen bekommt, irrt sich gewaltig. Der Amiga zählt nämlich auch Ä, Ü, Ö und andere Buchstaben des nicht englischen Sprachraums zu den Sonderzeichen. "Na und, solange er sie anerkennt, ist das doch egal. Oder?" Leider nicht.

Die meisten Programme (darunter CLI und Workbench) unterscheiden bei ÖÄÜ & Co nämlich zwischen Groß- und Kleinschreibung. In Filenamen kann dies unangenehme Folgen haben. Besonders, da das große Ü kaum vom kleinen zu unterscheiden ist. Auch bei Programmen, die alphabetisch sortieren, ist Vorsicht geboten. Sonderzeichen liegen nämlich sehr weit hinten im ASCII-Zeichensatz des Amiga.

2.3.6 Joker: Das DOS-Suchmuster

Bei vielen CLI-Befehlen (z.B. Copy) kann man mit sogenannten Jokern eine ganze Gruppe von Files auf einmal bearbeiten, z.B. kopieren. Im Gegensatz zu anderen Computern kann man mit den Jokern des Amiga auch komplexe Bedingungen formulieren. Das CLI betrachtet die folgenden Zeichen als Joker: ?, %, #, |, (und). Möchte man eines dieser Zeichen als Teil eines Dateinamens verwenden, muß man ihm ein Apostroph voranstellen, um es von einem Joker zu unterscheiden. Da es im Zeichensatz des Amiga verschiedene Zeichen gibt, die dem Apostroph sehr ähnlich sind, ist es wichtig zu wissen, wie man es nun auf der Tastatur erzeugt. Verwendet man die deutsche Tastaturbelegung, muß man [Alt] [Ä] drücken. Kommt in einem Filenamen ein Apostroph vor, müssen Sie ihm ein weiteres voranstellen, damit ein Unterschied zu dem Apostroph erkennbar wird, das einen Joker in ein normales Zeichen umdeklariert. Doch kommen wir zur Anwendung der Joker. Am bekanntesten ist die Kombination

```
#?
```

Sie setzt sich aus den Jokern # und ? zusammen. Das Doppelkreuz bedeutet, daß das Zeichen oder die Zeichengruppe hinter ihm beliebig oft wiederholt werden kann. In unserem Beispiel wäre das das Fragezeichen. Alternativ kann man das Fragezeichen auch durch

```
(?)
```

ersetzen. Die Klammern fassen dabei mehrere Zeichen zu einer Zeichengruppe zusammen. Mit anderen Worten entspricht (?) dem einfachen Fragezeichen, da nur ein Zeichen zwischen den Klammern steht. Dieses Fragezeichen ist nun wiederum ein Joker. Es steht für ein beliebiges Zeichen.

```
Delete #?
```

löscht z.B. alle Dateien (und leeren Verzeichnisse) im aktuellen Directory.

Die Klammern fassen Ausdrücke zusammen. Das |-Zeichen ermöglicht es, verschiedene Kriterien wahlweise anzugeben, von denen nur eins zutreffen muß. Dieses Zeichen erhält man ganz einfach durch die Tastenkombination [Shift]+[Backslash] ([Shift]+[\]). Der Backslash, der auf keinen Fall mit dem als Slash bezeichneten Schrägstrich zu verwechseln ist, befindet sich

links neben der [Backspace]-Taste. Nun fehlt nur noch der %-Joker. Er vertritt kein Zeichen. Wie bitte? Was soll denn das? Wie Sie etwas später sehen werden, kann sich dieser sogenannte Nullstring-Operator als sehr nützlich erweisen.

Nun folgt eine übersichtliche Aufstellung aller Joker.

Joker	Funktion
?	Ein beliebiges Zeichen
#X	Beliebige Wiederholung eines Ausdrucks
%	Keine Zeichen
X1|X2	Entweder X1 oder X2
(X)	Zusammenfassung bzw. Schachtelung

Sehen wir uns die Wirkung der einzelnen Joker an einigen Beispielen an:

Suchmuster	Beispiele
x?y	xYz x1z x_z ...
x#y	x xy xyy xyyy xyyyy ...
x#yz	xz xyz xyyz xyyyz ...
x#(yz)	x xyz xyzyz xyzyzyz ...
x(y|z)	xy xz
x(y|z|%)	x xy xz
#?.info	.info CLI-Manager.info Textomat.Info ...
?#?.info	CLI-Manager.info Textomat.Info ...
T#?	Turbo Turbo.info Turbo.bak Tel Tel.info ...

2.3.7 Abfragen im CLI

Möchte man in der StartUp-Sequence oder in einem anderen Batch-File eine Frage an den Anwender stellen, die dieser mit "Ja" oder "Nein" beantworten soll, kann man sich des CLI-Befehls Ask bedienen. Gibt es mehr als zwei mögliche Antworten, bietet sich ein kleiner Trick an, auf den wir in diesem Kapitel auch noch zu sprechen kommen. Doch beginnen wir mit Ask.

Ein Aufruf von Ask gestaltet sich denkbar einfach:

```
Ask "Text, der ausgegeben werden soll"
```

Der Befehl gibt den Text aus und wartet darauf, daß Sie entweder "Yes" oder "No" eingeben. Drücken Sie nur [Return], geht Ask davon aus, daß Sie sich für "No" entschieden haben. Alternativ kann man auch "Y" und "N" verwenden.

Nachdem Ask eine Antwort bekommen hat, wird die Batch-Datei weiter abgearbeitet. Um feststellen zu können, was eingegeben wurde, müssen wir die Befehle If, Else und EndIf einsetzen. Sie arbeiten ähnlich wie die (hoffentlich bekannten) gleichnamigen BASIC-Anweisungen. Nähere Informationen können Sie übrigens dem CLI-Handbuch entnehmen, das Commodore mit dem 500er ausliefert. Allgemein läuft die Abfrage so ab:

```
Ask "Are you sure? (Yes/No)"     ; Eingabe holen
If WARN                          ; Wenn Yes oder Y, dann...
 Skip Label_1                    ;    weiter bei Label_1
Else                             ; Wenn nicht, dann...
 Skip Label_2                    ;    weiter bei Label_2
EndIf                            ; Ende der Abfrage
Lab Label_1                      ; Einsprungspunkt für Yes
Echo "Label 1 angesprungen"      ; Text ausgeben
Skip Label_3                     ; Label_2 überspringen
Lab Label_2                      ; Einsprungspunkt für No
Echo "Label 2 angesprungen"      ; Text ausgeben
Lab Label_3                      ; Hilfslabel
```

Geben Sie Yes ein, werden die Befehle ausgeführt, die zwischen If und Else stehen. Im anderen Fall werden die Zeilen verwendet, die durch Else und EndIf eingeschlossen sind. Im oben abgebildeten Beispiel ist das je ein Sprungbefehl. Skip ermöglicht es, in einer Batch-Datei nach vorne zu springen. Der Befehl sucht nach einer Zeile, in der Lab, gefolgt von einem Label, steht. Hinter dem Lab-Befehl, dessen Label identisch mit dem Argument von Skip ist, wird die Ausführung dann fortgesetzt. Label könnte man übrigens auch als Sprungmarken bezeichnen.

Man kann die Anweisungen, die je nach gegebener Antwort ausgeführt werden sollen, auch direkt zwischen If, Else und EndIf setzen. Ein Batch-File, das nach diesem System arbeitet, könnte z.B. so aussehen:

```
Ask "Are you sure? (Yes/No)"     ; Eingabe holen
Echo ""                          ; Leerzeile ausgeben
If WARN                          ; Wenn Yes oder Y, dann...
 Echo "Sie haben Y, Yes oder"    ; Text ausgeben
 Echo "etwas ähnliches"
```

```
 Echo "eingegeben"
Else                              ; Wenn nicht, dann...
 Echo "Sie haben N, No oder"     ; anderen Text ausgeben
 Echo "etwas ähnliches"
 Echo "eingegeben, oder einfach"
 Echo "nur Return gedrückt."
EndIf                             ; Ende der If-Struktur
Echo ""                           ; Leerzeile ausgeben
```

Leider ist es mit Ask nicht ohne weiteres möglich, mit "Ja" zu anworten. Wen dies stört, der kann das folgende Batch-Programm einsetzen. Es kommt ganz ohne Ask aus.

```
Echo "Eingabemöglichkeiten: J, Ja, N, Nein"
Skip >NIL: ?
Lab N
Lab Nein
Echo "Sie haben N oder Nein gewählt"
Skip Ende
Lab J
Lab Ja
Echo "Sie haben J oder Ja gewählt"
Lab Ende
```

Auch das folgende Batch-File ist interessant:

```
Echo "Welches Computerl' hätten's denn gern?"
Echo "(Eingabemöglichkeiten: 1, 2, 3, Amiga, ST, VAX)
Echo "1 Amiga"
Echo "2 ST"
Echo "3 VAX (Großrechner)"
Echo ""
Skip >NIL: ?
Lab Amiga
Lab 1
Echo "Sie haben eine gute Wahl getroffen!"
Skip Ende
Lab ST
Lab 2
Echo "Bitte, wie Sie wünschen."
Skip Ende
Lab VAX
Lab 3
Echo "Sind Sie denn noch zu retten?"
Echo "Kaufen Sie sich doch lieber"
Echo "ein paar hundert Amigas mit 68030/68882-Karte!"
```

```
Echo "Das kostet Sie auch keinen"
Echo "Pfennig mehr. Und jeder Amiga leistet soviel"
Echo "wie eine VAX!"
Lab Ende
```

2.3.8 Disketten mit Auswahlmenü

Packt man viele Programme auf eine Start-Diskette, stellt sich die Frage, wie man diese am besten aufruft. Den Weg über die Workbench wollen wir einmal außer acht lassen. Unser Ziel ist es, ein System einzurichten, mit dem man kurze Zeit nach dem Booten über die Tastatur bereits das Programm wählen kann, das geladen werden soll.

Nehmen wir einmal an, wir hätten vier Programme: CLI-Manager, Disk-Manager, C-Monitor und AmigaBASIC. Zuerst soll ein Menü ausgegeben werden, das über die verfügbaren Programme informiert. Dann soll man eine Zahl eingeben, über die dann das zugehörige Programm geladen wird. Ein einfaches Verfahren ist es, die Programme direkt umzubenennen. Man könnte dann, nachdem man sich das Menü angesehen hat, die Kennziffer eingeben und damit das Programm laden.

Da wir jedoch keine halben Sachen machen wollen, überlegen wir uns, ob wir unser Problem nicht eleganter lösen können. Skip bietet sich dafür hervorragend an. Da es jedoch nicht gerade angenehm ist, für jede Diskette etliche Skip- und Lab-Befehle einzugeben, haben wir ein kleines BASIC-Programm geschrieben, das die nötige Kleinarbeit übernimmt. Lediglich die Programmnamen müssen noch eingegeben werden.

```
"************************************
"*                                  *
"*  Programm: StartUp-Generator.BAS *
"*                                  *
"*  Sprache:  AmigaBASIC            *
"*                                  *
"*  Datum:    22.02.1988            *
"*                                  *
"************************************
DIM Feld$(100)
COLOR 3
PRINT "StartUp-Menü-Generator"
COLOR 1
```

```
PRINT
LINE INPUT "Überschrift:",HeadLine$
CLS
PRINT "Geben Sie nun die Programmnamen ein, "
PRINT "die in die Liste"
PRINT "aufgenommen werden sollen. Drücken Sie nur Return,"
PRINT "wird die Eingabe abgeschlossen."
PRINT
Eingabe:
LINE INPUT Feld$(I%)
If Feld$(I%)<>"" THEN I%=I%+1: GOTO Eingabe
CLS
PRINT "Geben Sie den Namen der Text-Datei ein."
PRINT "Nur Return=Meldung.TXT"
INPUT Datei$
If Datei$="" THEN Datei$="Meldung.TXT"
CLS
PRINT "Geben Sie den Namen des Batch-Files ein."
PRINT "Nur Return=S/StartUp-Sequence"
INPUT Batch$
If Batch$="" THEN Batch$="S/StartUp-Sequence"
CLS
PRINT "In welchem Laufwerk befindet sich die Diskette?"
PRINT "DF";
I$=""
WHILE I$<"0" OR I$>"3"
   I$=INKEY$
WEND
Laufw$="DF"+I$+":"
PRINT I$
OPEN "o",#1,Laufw$+Datei$
   PRINT#1,CHR$(12); " CLS
   PRINT#1,HeadLine$
   PRINT#1,
   FOR II%=0 TO I%-1
      Help$=STR$(II%+1)
      Help$=RIGHT$(Help$,LEN(Help$)-1)
      If LEN(Help$)=1 THEN Help$=" "+Help$
      PRINT#1,Help$+" "+Feld$(II%)
   NEXT
   PRINT#1,
   PRINT#1,"Ihre Wahl:"
   PRINT#1,
CLOSE 1
OPEN "o",#1,Laufw$+Batch$
   PRINT#1,"System/SetMap d"
   PRINT#1,"Type ";Datei$
   PRINT#1,"Skip >NIL: ?"
```

```
   PRINT#1,
   FOR II%=0 TO I%-1
      PRINT#1,"Lab";II%+1
      PRINT#1,Feld$(II%)
      PRINT#1,"Skip Ende"
      PRINT#1,
   NEXT
   PRINT #1,"Lab Ende"
CLOSE 1
Run
```

Wenn Sie dieses Programm starten, werden verschiedene Eingaben von Ihnen verlangt. Zuerst ist die Überschrift an der Reihe, die über dem Menü angezeigt wird. Dann müssen die Namen der Programme eingegeben werden, die später aufgerufen werden können. Nach dem letzten müssen wir dann einfach [Return] drücken. Danach erkundigt sich das Programm nach dem Namen einer Meldungs-Datei, in die der Text geschrieben wird, der von der Batch-Datei ausgegeben wird. Diese Eingaben wird man meistens nicht von Hand vornehmen. Einfach nur zweimal [Return] drücken, dann werden die Standard-Werte verwendet. Schließlich ist noch einzugeben, in welchem Laufwerk die Diskette liegt, die hergerichtet werden soll. Wenn wir als Überschrift Tool-Diskette wählen und unsere vier Programme eingeben, erzeugt das Programm zwei Files: Meldung.TXT und S/StartUp-Sequence.

Inhalt von Meldung.TXT

```
Tool-Diskette

 1 CLI-Manager
 2 Disk-Manager
 3 C-Monitor
 4 AmigaBASIC

Ihre Wahl:
```

Inhalt von S/StartUp-Sequence

```
System/SetMap d
Type Meldung.TXT
Skip >NIL: ?
Lab 1
CLI-Manager
Skip Ende
Lab 2
```

```
Disk-Manager
Skip Ende
Lab 3
C-Monitor
Skip Ende
Lab 4
AmigaBASIC
Skip Ende
Lab Ende
```

2.3.9 Die richtige Arbeitsumgebung

Immer wenn es der Amiga für nötig hält, einen CLI-Befehl oder etwas anderes nachzuladen, werden wir aufgefordert, die Diskette, auf der der Amiga das gesuchte File vermutet, in "any drive", d.h. in ein beliebiges Laufwerk, einzulegen. Hat man nur ein einziges, wird der Diskettenwechsel oft zur Qual.

Um diesem Problem beizukommen, können wir alle Files, die oft nachgeladen werden, in der RAM-Disk ablegen. Mit einem speziellen CLI-Befehl (Assign) überlisten wir das Anwendungsprogramm bzw. das CLI, indem wir es dazu bringen, seine Files bzw. Befehle aus der RAM-Disk zu holen. Wir müssen allerdings eine Auswahl treffen, was in die RAM-Disk kopiert werden soll, da wir, wenn wir die komplette Workbench dorthin verlegen würden, nicht nur Zeit, sondern auch Speicher verschwenden, der, besonders wenn man nur 512 KByte hat, schnell voll werden kann.

Üblicherweise baut man die Anweisungen, die die CLI-Befehle in die RAM-Disk kopieren, in der StartUp-Sequence einer Startdiskette ein. Bootet man von ihr aus, wird auf die Befehle im RAM zugegriffen. Das folgende Listing verlegt die CLI-Befehle in die RAM-Disk, wenn es in die StartUp-Sequence eingebaut wird:

```
MakeDir RAM:c
Copy C:Copy RAM:c
Copy C:Dir RAM:c
Copy C:List RAM:c
.
.
.
Assign C: RAM:c
```

Bei diesem Verfahren können Sie die Befehle, die Sie oft verwenden und daher ins RAM kopieren wollen, selbst auswählen. Dieses Verfahren hat in der oben stehenden Variante jedoch einen kleinen Nachteil: Der Copy-Befehl muß jedesmal neu geladen werden. Im folgenden Listing umgehen wir dieses Problem, indem wir sofort, nachdem Copy kopiert wurde, mit Assign den weiteren Zugriff auf C: (das logische Verzeichnis, in dem sich die CLI-Befehle befinden) auf RAM:c umlegen. Alle nun folgenden Copy-Aufrufe holen sich den Befehl aus der RAM-Disk.

Leider gibt es aber noch ein weiteres Problem. Legt man das C:-Verzeichnis komplett auf die RAM-Disk um, können die Befehle, die sich nur auf der Diskette befinden, nicht mehr verwendet werden. Abhilfe schafft hier Path. Mit dem Path-Befehl können wir Pfade angeben, in denen nach CLI-Befehlen gesucht wird, wenn im aktuellem Verzeichnis bzw. im C:-Directory keine zu finden sind.

```
MakeDir RAM:c
Copy df0:c/Copy RAM:c
Copy df0:c/Path RAM:c
Assign C: RAM:c
Path df0:c
...
```

Eine komplette StartUp-Sequence könnte dann so aussehen:

```
System/SetMap d
AddBuffers DF0: 30
Stack 10000
MakeDir RAM:c
Copy df0:c/Copy RAM:c
Copy df0:c/Path RAM:c
Assign C: RAM:c
Path df0:c
Copy df0:c/Assign RAM:c
Copy df0:c/CD RAM:c
Copy df0:c/Delete RAM:c
Copy df0:c/Dir RAM:c
Copy df0:c/Ed RAM:c
Copy df0:c/EndCLI RAM:c
Copy df0:c/Execute RAM:c
Copy df0:c/Info RAM:c
Copy df0:c/Install RAM:c
Copy df0:c/List RAM:c
Copy df0:c/MakeDir RAM:c
```

```
Copy df0:c/NewCLI RAM:c
Copy df0:c/Protect RAM:c
Copy df0:c/Relabel RAM:c
Copy df0:c/Rename RAM:c
Copy df0:c/Run RAM:c
Copy df0:c/Type RAM:c
MakeDir RAM:l
Copy df0:l RAM:l
Assign L: RAM:l
LoadWB
EndCLI >NIL:
```

Neben einigen CLI-Befehlen, die uns besonders wichtig erschienen, kopieren wir auch das L:-Verzeichnis, in dem sich unter anderem auch der Disk-Validator befindet.

Auch haben wir die deutsche Tastaturbelegung eingestellt und den Stack, einen Speicherbereich, der Programmen für die Speicherung besonderer Daten zu Verfügung gestellt wird, vergrößert. Mit AddBuffers haben wir dem Laufwerk DF0: zusätzlich 30 Puffer zugeordnet, in die das Betriebssystem Daten zwischenspeichern kann. Unter Umständen kann man dadurch die Diskettengeschwindigkeit etwas steigern.

2.3.10 Steuerzeichen im Text

In normalen ASCII-Text-Files können wir neben direkt sichtbaren Zeichen noch sogenannte Steuer-Codes verwenden, die jeweils eine bestimmte Aufgabe erfüllen. Mit ihnen kann man z.B. den Inhalt eines CLI-Fensters löschen, die Textfarben und den Textstil ändern und noch vieles mehr. Wir haben für Sie eine kleine Auswahl möglicher Steuer-Codes zusammengestellt.

[Esc] spielt bei ihnen eine entscheidende Rolle. Man kann den Code der [Esc]-Taste nicht im Ed eingeben, da ihn dieser abfängt, um in den erweiterten Eingabe-, d.h. Befehlsmodus zu verzweigen. Gibt man eine Text-Datei jedoch über

```
Copy * Datei
```

ein, wird man mit diesem Problem nicht konfrontiert. Sehen wir uns doch einmal an, welche Steuerzeichen und Steuersequenzen (d.h. aus mehreren

Zeichen zusammengesetzte Ketten) es gibt. Steht in der folgenden Tabelle vor einem Buchstaben das Zeichen ^, muß dieser in Verbindung mit [Ctrl] gedrückt werden. Da auch dies nicht innerhalb des Ed möglich ist, haben wir für Sie ein kleines BASIC-Programm vorbereitet, das einen Text, in dem Sonderzeichen durch normale, aber selten verwendete Zeichen ersetzt wurden, in seine richtige Form umwandelt. Der umgewandelte Text kann dann mit Type im CLI-Fenster ausgegeben oder sogar an den Drucker (PRT:) geschickt werden. Doch sehen wir uns erst einmal ein paar Steueranweisungen an:

Code	Bezeichnung	Funktion	Hex-Code
^G	BEL	Piepser oder Blitzer	$07
^H	BS	[Backspace]	$08
^I	TAB oder HT	Horizontaler Tabulator	$09
^J	LF	Line Feed = Zeile runter	$0A
^K	VT	Vertikaler Tabulator	$0B
^L	FF	Form Feed = neue S. (CLS)	$0C
^M	CR	CRSR an Zeilenanfang	$0D
[Esc]	Esc	[Escape]	$1B

Auf der vorletzten Seite Ihres Amiga-500-Handbuches finden Sie noch weitere Steuersequenzen, die besonders für PRT: wichtig sind. "<CSI> ... m" bedarf einer genaueren Erklärung. Für ... kann man verschiedene Anweisungen angeben, die man mit einem Semikolon voneinander trennen muß. Diese Anweisungen betreffen die Farbwahl und den Schriftstil. Leider reagieren die unterschiedlichen Ausgabegeräte PRT: CON: und NEWCON: auf eine Anweisung manchmal mit verschiedenen Reaktionen. Hier hilft nur die "Trial and Error"-Methode: Ausprobieren, Fehler erkennen und nochmal ausprobieren. Farbcodes werden zweistellig angegeben, wobei die erste Ziffer entweder 3 (für die Zeichenfarbe) oder 4 für die Hintergrundfarbe sein kann. Danach folgt die Nummer der gewünschten Farbe. Die Schriftstile werden aus den folgenden Möglichkeiten zusammengesetzt:

Ziffer	Funktion
0	Normal
1	Fett
3	Kursiv
4	Unterstreichen
7	Invers

Die folgenden Steuersequenzen sind für die Geräte CON: RAW:, NEWCON:, * und bedingt für PRT: gültig. Bei der direkten Ausgabe auf PAR: sind sie meist sinnlos. Steht bei der Sequenz (n), muß dieser Ausdruck durch eine dezimale Zahl in ASCII-Schreibweise ersetzt werden. Gibt man keine Zahl an, geht AmigaDOS davon aus, daß Sie den Wert Eins wünschen. <CSI> vertritt den hexadezimalen Code $9B, der alternativ auch durch "<Esc>[" ersetzt werden kann.

Sequenz	Hexadezimal	Funktion
<CSI> (n) @	<CSI> (n) 40	(n) Leerzeichen an Cursorposition einfügen
<CSI> (n) A	<CSI> (n) 41	Cursor (n) Zeichen nach oben
<CSI> (n) B	<CSI> (n) 42	Cursor (n) Zeichen nach unten
<CSI> (n) C	<CSI> (n) 43	Cursor (n) Zeichen nach rechts
<CSI> (n) D	<CSI> (n) 44	Cursor (n) Zeichen nach links
<CSI> (n) E	<CSI> (n) 45	Cursor ganz nach links und (n) Zeichen nach unten
<CSI> (n) F	<CSI> (n) 46	Cursor ganz nach links und (n) Zeichen nach oben
<CSI> (n) H	<CSI> (n) 48	Cursor in Zeile (n) bringen (nur RAW:-Fenster)
<CSI> ;(n) H	<CSI> 3B (n) 48	Cursor in Spalte (n) bringen (nur RAW:-Fenster)
<CSI> (n1);(n2) H	<CSI> (n1) 3B (n2) 48	Cursor in Zeile (n1) und Spalte (n2) bringen (nur RAW:-Fenster)
<CSI> J	<CSI> 4A	Fenster ab Cursor nach unten löschen
<CSI> K	<CSI> 4B	Zeile ab Cursor nach rechts löschen
<CSI> (n) L	<CSI> (n) 4C	(n) Zeilen vor Cursorzeile einfügen
<CSI> (n) M	<CSI> (n) 4D	(n) Zeilen löschen; Zeilen unterhalb des gelöschten Bereichs werden nach oben geschoben.
<CSI> (n) P	<CSI> (n) 50	(n) Zeichen ab Cursorposition löschen
<CSI> (n) S	<CSI> (n) 53	Alles um (n) Zeilen nach oben schieben
<CSI> (n) T	<CSI> (n) 54	Alles um (n) Zeilen nach unten schieben
<CSI>0<SPACE>p	<CSI> 30 20 70	Cursor unsichtbar
<CSI> <SPACE>p	<CSI> 20 70	Cursor sichtbar
<CSI> ... m	<CSI> ... 6D	Farbe und Stil einstellen

Zum Schluß ein kleines Bespiel: Fett und Rot auf Schwarz wird gewünscht:

```
<CSI> 3;4;33;42m
```

Nun kommen wir endlich zu unserem angekündigten Programm, das Steuerzeichen in ASCII-Dateien einbaut. Ctrl-Codes werden im Quell-File mit einem vorangestellten ^-Zeichen gekennzeichnet. Möchte man ^ im Text verwenden, muß man $^ schreiben. [Esc] läßt sich durch ^[realisieren. Folglich sähe "<Esc>[" so aus: ^[[

Um hexadezimale Codes in den Text einzubauen, kann man das Dollarzeichen ($) verwenden. Hinter ihm muß dann die zweistellige (!) Hexadezimalzahl folgen. Wird das Dollarzeichen selbst im Text gewünscht, muß dies durch $$ gekennzeichnet werden.

Schauen Sie sich bitte erst einmal das Programm an:

```
"****************************
"**                        **
"**  Programm: SCodes.BAS  **
"**                        **
"**  Sprache:  AmigaBASIC  **
"**                        **
"**  Datum:    24.10.1988  **
"**                        **
"****************************

CLS
INPUT "Quell-Datei  <",Q$
INPUT "Ziel-Datei   >",Z$

OPEN "i",#1,Q$
OPEN "o",#2,Z$

WHILE EOF(1)=0
 LINE INPUT#1,A$
  B$=""
  FOR i%=1 TO LEN(A$)
   h$=MID$(A$,i%,1)
   If h$="^" THEN
    i%=i%+1
    If i%<= LEN(A$) THEN
     h$=MID$(A$,i%,1)
     If (ASC(h$) AND 127)>=64 AND (ASC(h$)
                                    AND 127)<64+32 THEN
```

```
      B$=B$+CHR$(ASC(h$)-64)
     Else
      B$=B$+h$
     END If
    END If
   ELSEIF h$="$" THEN
    i%=i%+1
    If i%<=LEN(A$) THEN
     h$=MID$(A$,i%,1)
     If i%+1<=LEN(A$) THEN
      i%=i%+1
      i$=MID$(A$,i%,1)
      h$=UCASE$(h$)
      i$=UCASE$(i$)
      If (h$>="0" AND h$<="9") OR (h$>="A"
                                   AND h$<="F") THEN
       If (h$>="0" AND h$<="9") OR (h$>="A"
                                   AND h$<="F") THEN
        j$="&h"+h$+i$
        B$=B$+CHR$(VAL(j$))
       END If
      END If
     Else
      B$=B$+h$
     END If
    END If
   Else
    B$=B$+h$
   END If
  NEXT
 PRINT#2,B$
WEND

CLOSE 2
CLOSE 1
```

Legen Sie bitte eine Text-Datei mit dem Namen RAM:Quell.ASC an. Sie soll den folgenden Inhalt erhalten:

```
^LDiese Zeile wird ganz oben ausgegeben
^[[33;42mROT auf SCHWARZ
$9B31;40m
```

Wenn Sie dieses File nun mit unserem Umwandlungsprogramm übersetzen und das Ergebnis in RAM:Ziel.ASC abspeichern, können Sie mit

```
Type RAM:Ziel.ASC
```

die schöne Ausgabe bewundern. Unser Prograämmchen eignet sich hervorragend dazu, übersichtliche Informations-Files zu PD- und Start-Disketten zu erstellen, die man von der StartUp-Sequence aus mit Type ausgeben kann. Denken Sie aber immer daran, den noch nicht übersetzten Quell-Code auf einer Diskette zu sichern, da Sie sonst Schwierigkeiten haben werden, das Text-File zu ändern.

2.3.11 Die unbekannte Sprache des Ed

Programmiersprachen gibt es wie Sand am Meer. Doch wußten Sie, daß es auch viele Editor-Programme für den Amiga gibt, die ebenfalls programmiert werden können, und daß einer dieser Editoren der Ed aus dem C-Verzeichnis der Workbench-Diskette ist?

Um es gleich vorwegzunehmen, die Programmiersprache des Ed ist nicht besonders leistungsfähig, da man hier nur eine einzige Programmzeile eingeben kann. Für einfache Anwendungen genügt sie jedoch.

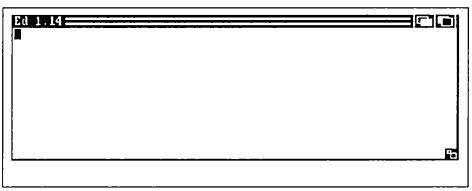


Abb. 2.4a: Der Workbench-Editor

Prinzipiell kennt Ed zwei Arten von Befehlen: direkte Befehle, die durch einen einzigen Tastendruck ausgeführt werden, und erweiterte Befehle, mit denen auch die Programme geschrieben werden. Um erweiterte Befehle eingeben zu können, muß man zunächst mit der Esc-Taste in den daür vorgesehenen Modus umschalten.

Nachdem man Ed gestartet hat, kann man Texte eingeben, mit dem Cursor durch den Text fahren und direkte Befehle eingeben. Welche direkten Befehle es gibt, können Sie der folgenden Aufstellung entnehmen:

Taste	Funktion
Del	Zeichen an Cursorposition löschen
Backspace	Zeichen links neben Cursor löschen
Tab	Cursor an nächste Tabulatorposition bewegen
Return	Zeile trennen bzw. neue Zeile einfügen
Esc	Erweiterte Befehle eingeben
Ctrl G	Wiederholung der zuletzt benutzten Befehlszeile
Ctrl A	Zeile hinter Cursorzeile einfügen
Ctrl B	Zeile löschen bzw. entfernen
Ctrl F	Groß-/Kleinschreibung unter Cursor tauschen
Ctrl U	Text schnell nach oben scrollen
Ctrl D	Text schnell nach unten scrollen
Ctrl R	Cursor auf Ende des vorhergehenden Wortes
Ctrl T	Cursor auf Anfang des nächsten Wortes
Ctrl E	Cursor an Fensteranfang oder -ende
Ctrl]	Cursor an Zeilenanfang oder -ende
Ctrl O	Wort oder Leerstellen bis Wort löschen
Ctrl Y	Von Cursor bis Zeilenende löschen
Ctrl H	Zeichen links neben Cursor löschen (Backspace)
Ctrl I	Cursor an nächste Tabulatormarke (Tab)
Ctrl M	Zeile trennen bzw. neue Zeile einfügen (Return)
Ctrl V	Bild neu aufbauen
Ctrl [	Erweiterten Modus aktivieren (Esc)

Nach den regulären kommen wir nun zu den erweiterten Befehlen. Bevor wir uns jedoch mit ihrer Anwendung beschäftigen, schauen wir uns erst einmal die Funktionen im Überblick an.

Befehl	Funktion
CL	Cursor ein Zeichen nach links
CR	Cursor ein Zeichen nach rechts
CS	Cursor an Zeilenanfang
CE	Cursor an Zeilenende
P	Cursor an Anfang der vorherigen Zeile
N	Cursor an Anfang der nächsten Zeile
T	Cursor an Textanfang
B	Cursor an Textende
M n	Cursor zu Zeile n bewegen
BS	Blockanfang markieren

Befehl	Funktion
BE	Blockende markieren
IB	Block hinter Cursorzeile einfügen
DB	Block löschen
SB	Block anzeigen
WB "Name"	Block unter "Name" abspeichern
If "Name"	Text-File "Name" hinter Cursorzeile einfügen
SA "Name"	Datei unter "Name" abspeichern. Wird "Name" nicht angegeben, wird der zuletzt benutzte Name (meist der, über den das File geladen wurde) verwendet.
X	Ed verlassen und Text speichern
Q	Ed nach Sicherheitsabfrage (y/n) verlassen, ohne Text zu speichern
F "Text"	Ab Cursorposition nach "Text" suchen
BF "Text"	Wie F, nur rückwärts
E "Tx1"Tx2"	"Tx1" einmal durch "Tx2" ersetzen
EQ "Tx1"Tx2"	Wie E, jedoch mit Sicherheitsabfrage (y/n)
LC	Bei den Befehlen F, BF, E und EQ wird beim Suchtext zwischen Groß- und Kleinschreibung unterschieden
UC	Im Gegensatz zu LC wird nach UC nicht mehr zwischen Groß- und Kleinschreibung unterschieden
S	Zeile an Cursorposition aufteilen
J	Zeile mit nachfolgender Zeile verbinden
A "Text"	"Text" wird nach der aktuellen Zeile eingefügt
I "Text"	"Text" wird vor der aktuellen Zeile eingefügt
D	Zeile löschen
DC	Zeichen an Cursorposition löschen
U	Undo-Funktion: Alle Änderungen, die in der aktuellen Zeile vorgenommen worden sind, werden, falls der Cursor nicht zwischenzeitig in eine andere Zeile bewegt wurde, rückgängig gemacht.
SH	Informationen über freien Speicher, markierten Block und Filename anzeigen
SL n	Linken Rand auf Spalte n setzen
SR n	Rechten Rand auf n setzen
ST n	Tabulatorabstand auf n einstellen
EX	Zeilenumbruch für aktuelle Zeile ausschalten

Befehl	Funktion
RP	Nachfolgenden Befehl ausführen, bis Fehler auftritt. Folgt ein eingeklammerter Ausdruck, werden alle Befehle zwischen den Klammern ausgeführt.

Zur obigen Tabelle müssen wir, um Mißverständnisse zu vermeiden, noch einige Anmerkungen machen. So kann man Textausdrücke z.B. nicht nur mit Anführungs-, sondern auch mit jedem beliebigen anderen Sonderzeichen begrenzen, wenn man das Anführungszeichen im Text verwenden will. Das gilt übrigens auch für File-Namen. Der E-Befehl, der eine Zeichenfolge durch eine andere ersetzt, hat ein etwas ungewöhnliches Format. Normalerweise gibt man zwei Parameter, durch eine Leerstelle getrennt, an. Bei diesem Befehl verwendet man zur Trennung der beiden Texte jedoch nur das Zeichen, mit dem die ersten Zeichenkette eingeleitet wurde. Beispiele:

```
E "Text1"Text2"
E /Text1/Text2/
```

Aber nicht:

```
E "Text1" "Text2"
```

Der E-Befehl wird nicht nur zum Ersetzen, sondern auch zum Löschen und Einfügen von Texten verwendet. Möchte man eine bestimmte Zeichenfolge löschen, gibt man diese als ersten Parameter des Befehls an. Beim zweiten übergibt man einen Leerstring. Bei den folgenden Beispielen wird das Wort "Test" gelöscht:

```
E .Test..
E "Test""
```

Möchte man an der aktuellen Cursorposition einen Text einfügen, kann man sich ebenfalls des E-Befehls bedienen. Der einzufügende Text wird als zweiter Parameter übergeben. Diesmal wird der erste Parameter jedoch mit einem Leerstring abgefertigt. Der folgende Befehl, der nur in Programmen sinnvoll ist, da es sonst schneller geht, den Text von Hand einzufügen, fügt an der Stelle, an der der Cursor steht, das Wort "Amiga" ein:

```
E ""Amiga"
```

Möchte man im erweiterten Eingabemodus mehrere Befehle eingeben, muß man diese durch ein Semikolon trennen. So bewirkt z.B. der folgende Aufruf, daß der Cursor zunächst zwei Zeilen nach unten und dann drei Zeichen nach rechts bewegt wird.

```
N;N;CR;CR;CR
```

Wenn Sie den Cursor nicht zwei, sondern 20 Zeilen nach unten bewegen wollen, ist es nicht sinnvoll, 20 mal den N-Befehl und das Semikolon einzugeben. Man kann daher vor einen Befehl eine Zahl setzen, die angibt, wie oft der Befehl ausgeführt werden soll.

Hat man einen Fehler gemacht, kann man die Ausführung der erweiterten Befehlszeile jederzeit mit der [Esc]-Taste abbrechen.

Soll nicht ein einzelner Befehl, sondern eine ganze Reihe von Anweisungen mehrmals ausgeführt werden, muß man sich eines weiteren Mittels bedienen: der Klammern. Die Befehle, die man mehrmals aufrufen möchte, werden durch runde Klammern eingeschlossen. Vor die öffnende Klammer kann dann die Zahl der Wiederhohlungen gesetzt werden. Auch kann man hier den RP-Befehl verwenden, der den folgenden Befehl oder die folgende, durch Klammern eingeschlossene Befehlsgruppe so oft wiederholt, bis ein Fehler auftritt. RP findet immer dann Anwendung, wenn man ein komplettes Text-File bearbeiten will. Doch sehen wir uns zunächst ein einfaches Beispiel für die Klammerung an. Es bewirkt, daß in einem regelmäßigen Abstand ein Zeichen ausgegeben wird. Wenngleich dies keine besonders sinnvolle Anwendung ist, demonstriert sie doch gut die Funktionsweise der Klammerfunktion.

```
CS;10(E""*";3CR)
```

Nachdem die theoretischen Grundlagen weitgehend klar sind, können wir uns einer praktischen und sinnvollen Anwendung zuwenden. Als Beispiel dient uns hier die Programmiersprache AmigaBASIC. Während der Arbeit an einem Programm ist es oft nötig, Zwischenergebnisse auszugeben, die später, wenn das Programm fertig ist, nicht mehr wichtig sind. Wenn man nun, während man an dem Programm arbeitet, alle Zeilen markiert, die nur zu Test-Zwecken wichtig sind, kann man den Ed dazu verwenden, aus der Test-Version des Programms eine endgültige Version zu machen, die sich auch zur Weitergabe eignet. Den alten Quellcode hebt man natürlich noch auf, da er sich gut dazu eignet, Veränderungen vorzunehmen.

Bevor man mit der Erstellung des Programms beginnt, überlegt man sich eine seltene Zeichenkombination, die an keiner anderen Stelle vorkommen wird. Diese Zeichenfolge sollte mit dem Apostroph beginnen, da dieses Zeichen in BASIC einen Kommentar einleitet und somit verhindert wird, daß die Markierung einen Syntax-Error produziert. Angenommen, die Erkennungszeichenfolge sei auf

```
"Help
```

festgelegt. Wenn wir nun ein Programm geschrieben haben, in dem hinter jeder Zeile, die nur bei Test-Läufen benutzt werden soll, die oben genannte Kombination steht, können wir dieses Programm mit

```
Save "Name",a
```

abspeichern. Wichtig ist dabei das ",a", welches dafür sorgt, daß das File nicht im speziellen, nur für AmigaBASIC verständlichen Format, sondern im Klartext gespeichert wird. Dieses File können wir nun mit dem Ed laden, indem wir im CLI

```
Ed Name
```

eingeben. Nun müßte der Programmtext angezeigt werden. Wenn wir jetzt die Esc-Taste drücken, zeigt Ed in der untersten Zeile des Fensters ein Sternchen, das zur Eingabe erweiterter Befehle auffordert. Dort geben wir

```
T;UC;RP(F."help.;D)
```

ein. Diese Anweisung führt genau das aus, was wir wollten: Alle Zeilen, die unsere Markierung enthalten, werden gelöscht. Dazu sind wir Ihnen mal wieder eine kleine Erklärung schuldig. Die Befehlszeile beginnt mit T. Dadurch wird der Cursor an den Text-Anfang gesetzt. Mit UC sorgen wir dafür, daß Ed nicht zwischen Groß- und Kleinschreibung unterscheidet. Das ist deshalb wichtig, damit die Kennung ("Help), unabhänig davon, ob sie groß, klein oder gemischt geschrieben wird, von der Such-Funktion (F-Befehl) erkannt wird.

Nun kommen wir zu RP. Hinter diesem Befehl steht ein Klammerausdruck, der so lange wiederholt wird, bis ein Fehler auftritt. Dies ist dann der Fall, wenn einer der Befehle in der Klammer einen solchen produziert. Dabei interessiert uns zunächst F. Dieser Befehl liefert erst dann eine Fehlermeldung, wenn alle Markierungen abgearbeitet worden sind.

Nachdem eine unserer Makierungen gefunden wurde, steht der Cursor in der betreffenden Zeile, die dann mit D komplett gelöscht wird.

Schließlich müssen wir Sie leider noch auf ein Problem hinweisen: Ed ist nicht ganz fehlerfrei. Es kann also sein, daß der Editor auf bestimmte Befehlskombinationen allergisch reagiert und Unsinn fabriziert.

2.3.12 Highlights des CLI

Die Shell

Die Shell unterscheidet sich auf den ersten Blick nicht vom normalem CLI. Untersucht man sie jedoch genauer, wird man unter anderem feststellen, daß nun endlich Korrekturen innerhalb der Zeile möglich sind. Auch kann man eingegebene Anweisungen nochmals aufgreifen und sich auf diese Weise viel Tipparbeit sparen.

Die Shell kann über ein Icon im Hauptverzeichnis der Workbench-Diskette oder über den Befehl

```
NewShell
```

aufgerufen werden. Gibt man diesen Befehl ein, wird automatisch die Batch-Datei "S:Shell-Startup" aufgerufen, in der z.B. der Prompt definiert werden kann. Auch für das normale CLI gibt es jetzt eine Batch-Datei, die immer dann ausgeführt wird, wenn ein neues CLI-Fenster geöffnet und kein besonderes Batch-File angegeben wird. Sie heißt "S:CLI-Startup".

Um alle neuen Funktionen des CLI bzw. der Shell ausnutzen zu können, muß die Anweisung

```
Resident CLI L:Shell-Seg SYSTEM pure add
```

bereits aufgerufen worden sein. Dies geschieht meistens in der Startup-Sequence.

Ist die Shell geladen, kann man ganz normal arbeiten wie im CLI. Hat man sich am Anfang einer Zeile vertippt, fährt man die fehlerhafte Stelle einfach

mit dem Cursor an, um diese zu korrigieren. Zu unserer großen Freude funktioniert jetzt auch die [Del]-Taste richtig. [Backspace] kann natürlich auch weiterhin verwendet werden.

Hat man seinen Tippfehler nicht bemerkt, muß man sich keineswegs die Haare raufen, weil man alles nochmals eintippen muß. Einfach [↑] drücken, und der falsch eingegebene Befehl kann geändert werden. Wollen Sie eines der vorhergehenden Kommandos erneut aufrufen, drücken Sie so oft [↑], bis es angezeigt wird. Mit [↓] können Sie den Befehlspuffer auch in die andere Richtung durchblättern.

Mit [Shift] [←] oder [Ctrl] [A] gelangen Sie an den Anfang der aktuellen Zeilen. Mit [Shift] [→] oder [Ctrl] Z erreichen Sie genau das Gegenteil: der Cursor wird hinter das letzte Zeichen gesetzt. [Ctrl] [X] löscht die komplette Zeile. Mit [Ctrl] [K] bzw. [Ctrl] [U] ist es nun aber möglich, den Text rechts bzw. links neben dem Cursor zu löschen.

Effektiv: Der Resident-Befehl

Die Tatsache, daß alle CLI-Befehle (außer Alias) nachgeladen werden müssen, hat, wie Sie sicher schon oft zu Ihrem Leidwesen feststellen mußten, nicht nur Vor-, sondern auch Nachteile. Die Klagen darüber sind sogar bis zu den Commodore-Entwicklern vorgedrungen, die dann, wenn auch erst ziemlich spät, einen Befehl entwickelt haben, mit dem man CLI-Befehle und bestimmte Programme im Speicher halten kann. Dies geht sehr einfach. Man sollte, wenn möglich, den Resident-Befehl der RAM-Disk-Methode vorziehen. Greifen nämlich mehrere Tasks auf ein Programm zu, beansprucht es, wenn es mit Resident geladen wurde, nur einmal den nötigen Speicherplatz. Die Daten werden natürlich für jeden Task getrennt verwaltet. Leider kann man nicht alle Programme resident machen. Die CLI-Befehle der Workbench 1.3 sind dafür jedoch wie geschaffen. Auch andere Tools werden nach und nach den Bedürfnissen angepaßt werden.

Normalerweise sollten alle Programme, die resident gehalten werden können, mit einem speziellen Flag gekennzeichnet sein. Da dies aber in den seltensten Fällen so ist, kann man Resident mit der Option Pure dazu bewegen, auch diese Programme zu akzeptieren. Routinen, die das Betriebssystem benötigt, müssen beim Aufruf des Resident-Befehls mit SYSTEM gekennzeichnet wer-

den. Möchte man einen residenten Befehl aus dem Speicher entfernen, hilft die Option Remove weiter. Mit Replace kann man ein anderes Programm unter einem schon bestehenden Namen ablegen und damit den Vorgänger aus dem Speicher werfen.

```
Resident C:Ed Pure
```

lädt z.B. den Ed als residentes Programm. Wenn Sie nun

```
Ed FileName
```

eingeben, sucht AmigaDOS gar nicht auf Diskette, sondern ruft gleich den schon im Speicher befindlichen Ed auf.

Man kann einen Befehl sogar unter einem anderen Namen installieren. Auch hier gibt es wieder ein Beispiel:

```
Resident Ex C:Execute Pure
```

Wenn Sie nun Ex eingeben, ruft AmigaDOS den Execute-Befehl auf.

Praktisch: Der neue List-Befehl

Auch der List-Befehl wurde erheblich verbessert. Leider können wir Ihnen nicht alle Veränderungen aufzählen, die an List und anderen Befehlen vorgenommen wurden, da der Stoff geeignet ist, ein eigenes Buch zu füllen. Doch kommen wir zurück zum List-Befehl. Angenehm ist, daß es nun auch möglich ist, ein DOS-Suchmuster direkt ohne S bzw. P anzugeben. Die wesentlichste Änderung liegt jedoch in einem neuen Schlüsselwort: LFORMAT.

Mit LFORMAT ist es möglich, einzustellen, in welcher Aufmachung die einzelnen Namen ausgegeben werden sollen. Diese Möglichkeit wird erst bei näherem Hinsehen interessant. Man kann das Format nämlich so festlegen, daß die Ausgabe des List-Befehls als Batch-Programm interpretiert werden kann. Demonstration gefällig? Gerne! Geben Sie doch einmal

```
CD SYS:
List LFORMAT="SetDate %s 01-Jan-88 00:00" TO
RAM:Batch
Type RAM:Batch
```

ein. Als Ausgabe erhalten Sie

```
SetDate Expansion.info 01-Jan-88 00:00
SetDate Trashcan 01-Jan-88 00:00
SetDate c 01-Jan-88 00:00
SetDate Prefs 01-Jan-88 00:00
SetDate System 01-Jan-88 00:00
SetDate l 01-Jan-88 00:00
SetDate Shell 01-Jan-88 00:00
usw.
```

Wenn Sie nun

```
Execute RAM:Batch
```

eingeben, wird das Datum aller Files in SYS: geändert. Möchte man nur bestimmte Files bearbeiten, kann man mit

```
Ed RAM:Batch
```

die Befehlsdatei noch überarbeiten. Man kann List natürlich auch ein Suchmuster mit auf den Weg geben, das dann automatisch die richtigen Files auswählt.

Hinter LFORMAT haben wir innerhalb der in Anführungszeichen stehenden Zeichenkette %s verwendet. Wie unschwer zu erkennen war, ersetzte List diese Kennung durch den aktuellen File-Namen. Verwendet man %s jedoch mehrmals, treten unterschiedliche Sonderfälle ein. Verwendet man %s zweimal, wird auch zweimal der File-Name eingesetzt. Anders sieht es allerdings aus, wenn man die Kennung gleich dreimal verwendet. Hierbei werden beim ersten der Pfad und beim zweiten und dritten der Eintragsname verwendet. Setzt man %s sogar viermal ein, werden das jeweils erste und dritte durch den Pfad, das jeweils zweite und vierte durch den Eintrag ersetzt. Diese Einteilung ist sehr nützlich. Möchte man z.B. veranlassen, daß bestimmte Files in ein anderes Verzeichnis verschoben werden, hilft das folgende Beispiel weiter, das sich des Rename-Tricks bedient.

```
List RAM:TEST1/#?.info LFORMAT="Rename %s%s RAM:TEST2/%s"
    TO RAM:Batch
```

Das Ergebnis, das in RAM:Batch in Form eines durch Execute ausführbaren Programms abgelegt wird, könnte dann z.B. so aussehen:

```
Rename RAM:TEST1/More.info RAM:TEST2/More.info
Rename RAM:TEST1/Clock.info RAM:TEST2/Clock.info
Rename RAM:TEST1/Notepad.info RAM:TEST2/Notepad.info
```

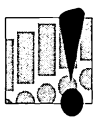
Trickreich: Der neue Format-Befehl

Ruft man Format vom CLI (oder der Shell) auf, hat man die Möglichkeit, eine Diskette, die schon einmal formatiert wurde, in wenigen Sekunden zu löschen. Der Trick bei der Sache liegt darin, nur die Teile der Diskette zu löschen, die unbedingt "leer" sein müssen, damit im Inhaltsverzeichnis keine Einträge stehen. Wen's interessiert: Format löscht den Root-Block, schreibt einen neuen Bitmap-Block und initialisiert den Boot-Block so, daß mit der Diskette nicht gebootet werden kann. Der größte Teil der Daten wird also nicht gelöscht, sondern nur zum Überschreiben freigegeben. Es versteht sich von selbst, daß dieses Verfahren auf Disketten mit Read-Errors (Lesefehlern) nicht angewendet werden kann. Solche Disketten müssen auf die normale Art formatiert werden. Klappt auch das nicht, kann man die Diskette ausrangieren.

Um Format klar zu machen, daß wir die Schnellformatierung wünschen, hängen wir an die üblichen Parameter noch QUICK an. Das könnte z.B. so aussehen:

```
Format DRIVE DF0: NAME Amiga_500_Buch QUICK
```

Allen, denen es noch nicht bekannt ist, verraten wir hiermit, daß man Format auch dazu bringen kann, den lästigen Mülleimer nicht zu installieren. Für diesen Zweck wurde das Schlüsselwort NOICONS reserviert.



Wenn beim Formatieren einer Diskette ein schleifendes Geräusch auftritt, das durch ein ungewöhnliches metallisches Knacken begleitet wird, können Sie damit rechnen, daß Ihnen diese Diskette in absehbarer Zeit mit einem Lesefehler aufwarten wird. Es ist ratsam, sie nicht für wichtige Daten zu verwenden. Tritt besagtes Geräusch bei einem anderen Diskettenzugriff (z.B. Laden) auf, weist es ebenfalls auf einen anstehenden Fehler hin. Dieser muß zwar nicht in den nächsten Monaten auftreten, wird sich jedoch über kurz oder lang bemerkbar machen.

Gegen Viren: Der neue Install-Befehl

Auch am Install-Befehl hat man lang ersehnte Änderungen vorgenommen. So ist es z.B. möglich, aus einer bootfähigen Diskette eine Daten-Diskette, d.h. eine solche zu machen, von der man nicht booten kann.

```
Install DF0: NOBOOT
```

Weiterhin kann man einen bestehenden Boot-Block auch auf dessen Inhalt untersuchen, um festzustellen, ob sich dort vielleicht ein Virus eingenistet hat. Dazu müssen wir

```
Install DF0: CHECK
```

eingeben. Sollten Sie nur ein Laufwerk besitzen, müssen Sie sich mit dem Fragezeichen oder der RAM-Disk behelfen.

Wo sind wir denn? - Der Pfad im Prompt

Im CLI oder der neuen Shell wird immer dann, wenn der Amiga eine Eingabe erwartet, der sogenannte Prompt angezeigt. Dieser besteht aus der Task-Nummer, gefolgt vom aktuellen, mit CD eingestellten Pfad. Auf diese Weise ist man immer informiert, wo man sich gerade auf einer Diskette befindet. Diese Einstellung ist schon auf der Workbench-Diskette vorgenommen worden. Es kann aber nichts schaden, zu wissen, wie dies realisiert werden kann.

```
Prompt "Task=%n  Pfad=%s  -->"
```

Für %n setzt die Shell die Task-Nummer und für %s den aktuellen Pfad ein. Übrigens kann man den Prompt auch farbig gestalten. Dazu baut man einfach die Codes zur Farbeinstellung ein. Allerdings darf man nicht vergessen, nach dem eigentlichen Prompt die Ausgabe wieder in den Normal-Zustand zurück zu versetzen, da sonst auch alle Ausgaben im CLI in der anderen Farbe wiedergegeben würden. [Esc] wird innerhalb der Anführungszeichen bei Prompt übrigens durch *e ersetzt. Der folgende Prompt wird in roter Farbe ausgegeben:

```
Prompt "*e[33;1mTask=%n  Pfad=%s  -->*[31;0m"
```

Jetzt möglich: Batch-Programme über Workbench aufrufen

Leider war es bisher nicht ohne weiteres möglich, Batch-Programme über die Benutzeroberfläche zu starten. Das ist jetzt anders. Durch das Programm IconX aus dem C-Verzeichnis sind wir nun in der Lage, durch das Anklicken eines Icons ein CLI-Fenster zu öffnen, in dem ein bestimmtes Batch-Programm ausgeführt wird. Das Batch-Programm muß dazu mit einem Projekt-Icon ausgestattet werden, dessen Default Tool auf C:IconX eingestellt ist.

Klickt man dieses Icon an, wird das Prográmmchen IconX aufgerufen, das ein CLI öffnet und darin die zum Icon gehörende Stapel- (= Batch-) Datei ausführt. Diese Technik wird auf der Workbench-Diskette bei dem File Copy-Prefs aus dem Prefs-Ordner angewendet. Sie können dort etwas "abgucken", falls die eigene Kreation nicht auf Anhieb klappt.

Kurz und bündig: Abkürzungen mit Alias

Hat man L:Shell-Seg resident geladen, kann man innerhalb eines CLI- oder Shell-Tasks Abkürzungen definieren, die man alternativ zu den normalen Befehlen eingeben kann. Die dazu nötigen Definitionen sollte man am besten in S:Shell-Startup einbauen, damit man immer, wenn man eine neue Shell öffnet, auf sie zugreifen kann. Beispiel gefällig? Bitte sehr:

```
Alias EdS Ed S:Startup-Sequence
```

Wenn Sie nun

```
EdS
```

eingeben, wird eigentlich

```
Ed S:Startup-Sequence
```

ausgeführt. Alias gestattet es sogar, hinter der Abkürzung noch Parameter anzugeben, die dann den Text, der durch das Kürzel ersetzt wird, anhängt. Sollen die Parameter an einer anderen Stelle eingefügt werden, macht man dies durch [] deutlich:

```
Alias Hex Type [] hex
```

Wenn Sie nun

```
Hex Test-File
```

eingeben, führt die Shell den Aufruf

```
Type Test-File hex
```

aus.

Kommunikation groß geschrieben: Das PIPE-Device

Mit dem PIPE-Device steht dem Amiga nun ein einfach zu handhabender Datenkanal zwischen Programmen zur Verfügung. Auf den Punkt gebracht, bedeutet das, daß nun jedes Programm Daten über einen Filenamen, der PIPE: als Gerät enthält, in einen Puffer schicken kann, der von einem anderen Programm, das den selben Filenamen verwendet, ausgelesen werden kann. Anhand einiger Beispiele wollen wir dieses an UNIX angelehnte Verfahren einmal ausprobieren. Dazu müssen wir PIPE: zunächst einbinden. Dies geschieht mit der Anweisung

```
Mount PIPE:
```

Denken Sie bitte daran, daß dieser Befehl auf die Datei DEVS:MountList zugreift. Je nach Workbench-Version kann es sein, daß PIPE: schon von der Startup-Sequence aus eingebunden wurde und daher sofort zu Verfügung steht.

Nun öffnen wir drei CLI-Fenster. Im ersten geben wir

```
Dir >PIPE:Kanal_1 SYS:
```

und im zweiten

```
List >PIPE:Kanal_2 SYS:
```

ein. Und was passiert? Nichts! Jedenfalls sehen wir nichts. Die Tatsache, daß das Laufwerk, in dem die System-Diskette liegt, anläuft, verrät uns jedoch, daß die beiden Befehle ausgeführt werden. Nur die Ausgabe scheint ähnlich

wie bei NIL: ins Leere zu gehen. Dem ist jedoch nicht so. Wenn wir nun im dritten CLI nacheinander

```
Type PIPE:Kanal_1
```

und

```
Type PIPE:Kanal_2
```

eingeben, können wir die Ausgaben, die in den beiden anderen CLI erzeugt wurden, abrufen. In diesem Fall wäre es zwar einfacher gewesen, ohne PIPE zu arbeiten, wenn man jedoch komplexere Anweisungen verwendet, kann PIPE: sehr nützlich sein. So kann man z.B. ein Inhaltsverzeichnis als Hex-Dump ausgeben:

```
Type PIPE:Kanal_1 hex
```

Wenn Sie diese Anweisung eingeben und sich wundern, daß nichts geschieht, liegt dies an einer Eigenart des PIPE-Device. Auch wenn es wie eine normale Diskette (abgesehen von Unterverzeichnissen) angesprochen wird, so geht es mit den Daten doch völlig anders um. Was über einen PIPE-Kanal ausgegeben wird, kann nur einmal gelesen werden. Wenn Sie das Programm abbrechen, das Daten aus einem Kanal liest, sind übrigens alle weiteren Daten, die ein anderer Task in diesen Kanal schreibt, verloren. Es ist also nicht möglich, nach ^C und erneutem Type die Daten weiter auszulesen.

Ein großer Vorteil von PIPE: ist es, daß Daten nun endlich ohne einen zwischenzeitigen Zugriff auf ein Speichermedium übermittelt werden können. Die Vorteile liegen klar auf der Hand: Zunächst spart man Zeit, da der interne Weg über den RAM-Speicher viel schneller ist als ein Zugriff auf ein Laufwerk. Auch die RAM-Disk kann mit PIPE: nicht mithalten. Denn auch sie muß die zu übermittelnden Daten kurzzeitig komplett aufnehmen, was nicht nur Speicherplatz kostet, sondern auch eine lästige Dateileiche produziert, die mit Delete wieder entfernt werden muß.

PIPE: kommt mit wenigen KByte Speicher aus, da das sendende Programm so lange warten muß, bis der Empfänger die Daten aus den Puffer entnommen hat und dieser für neue Informationen frei ist.

Schließlich sei noch auf ein wirklich sinnvolles Anwendungsbeispiel verwiesen: Erinnern Sie sich noch an den neuen List-Befehl, der mit LFORMAT zur Produktion von Batch-Dateien verwendet werden konnte?

Es liegt nun auf der Hand, in einem CLI den List-Befehl zu starten und dabei die Ausgabe auf einen Datenkanal umzulegen. Vom einem anderen Task kann dieser nun über den Execute-Befehl ausgelesen werden.

Im ersten CLI ist einzugeben:

```
List >PIPE:Kanalname LFormat="SetDate %s 01-Jan-89
00:00:00"
```

Im zweiten CLI ist einzugeben:

```
Execute PIPE:Kanalname
```

Da es nicht gerade angenehm ist, immer mit mehreren CLI-Fenstern zu arbeiten, ist es ratsam, den Run-Befehl zu verwenden, wenn PIPE: eingesetzt wird. Im obigen Beispiel hätte man z.B. den List-Befehl mit Run starten können. Auf diese Weise käme man ohne ein zweites CLI aus.

Reset-fest: Das RAD-Device

Die reset-feste RAM-Disk wird über den Gerätenamen RAD: angesprochen und mit Mount aktiviert. Im Gegensatz zu RAM: wird sie nicht dynamisch verwaltet, was zunächst einmal ein ziemlich großer Nachteil ist, da immer eine bestimmte Speichermenge beansprucht wird, auch wenn sich keine oder nur wenige Files in RAD: befinden. Man hat sich jedoch für dieses Verfahren entschieden, um im Falle eines Systemabsturzes die Wahrscheinlichkeit, daß ein Teil der Daten in der RAM-Disk überschrieben wird, möglichst gering zu halten.

Bevor wir mit dem Befehl

```
Mount RAD:
```

den Zugriff auf das neue Gerät ermöglichen, müssen wir uns noch überlegen, wieviel Speicher wir der RAM-Disk zugestehen wollen. Dazu sehen wir uns den Inhalt der MountList mit

```
Ed Devs:MountList
```

an. Dort müßte sich unter anderem der folgende Eintrag befinden:

```
RAD:        Device = ramdrive.device
            Unit   = 0
            Flags  = 0
            Surfaces  = 2
            BlocksPerTrack = 11
            Reserved = 2
            Interleave = 0
            LowCyl = 0  ;  HighCyl = 21
            Buffers = 5
            BufMemType = 1
#
```

Dieser Eintrag enthält alle Informationen, die das Betriebssystem benötigt, um RAD: zu öffnen. Wie sie sehen, wird dazu das File "Devs:ramdrive.device" nachgeladen.

Was uns jedoch an diesem Eintrag interessiert, ist die Speicherkapazität, die die RAM-Disk erhalten soll. Sie hängt von den Einträgen Surfaces, BlocksPerTrack, LowCyl und HighCyl ab. Sinnvoll ist es, die folgenden Werte nicht zu verändern:

```
Surfaces  = 2
BlocksPerTrack = 11
LowCyl = 0
```

Die Speicherkapazität hängt jetzt nur noch von HighCyl ab. Ein Plus oder Minus von 1 entspricht real 11 KByte. In der obigen MountList-Einstellung wird also ein Speicher von 242 KByte belegt. Sollten sie den Wert 231 errechnen, haben sie nicht beachtet, daß nicht 21 sondern 22 Spuren belegt werden. Es gibt nämlich auch eine Spur mit der Nummer Null.

242 KByte sind nun aber nicht gerade wenig. Besonders, wenn man nur 512 KByte hat, ist dies unnötig viel. Um Ihnen die Rechnerei zu ersparen, haben wir einige Werte aufgeschrieben:

HighCyl	Speicherkapazität in KByte
5	66
10	121
15	176
20	231
25	286

Wir warnen jedoch vor haarscharfen Kalkulationen. Wollen Sie beispielsweise ein File, das eine Länge von genau 66 KByte hat, in der RAD-Disk (so wollen wir RAD: ab jetzt nennen) ablegen, werden Sie, wenn Sie ihr nur 66 KByte zugestanden haben, über einen Requester darauf hingewiesen, daß der Speicher der RAD: aufgebraucht ist. Dies liegt daran, daß neben den eigentlichen Daten noch einige Zusatzinformationen gespeichert werden (z.B. Filename).

Nach der Theorie wenden wir uns nun der Praxis zu. Geben Sie, nachdem Sie Ed verlassen haben, den schon genannten Befehl

```
Mount RAD:
```

ein. Prinzipiell könnten wir die RAD-Disk zwar schon verwenden, sinnvoll ist es jedoch, sie zu formatieren, damit sie auch von der Workbench angezeigt wird. Bei der uns vorliegenden Workbench-Version war das zwar nicht zwingend notwendig, es kann jedoch sein, daß nachfolgende Workbench-Disketten dies verlangen. Wir geben also ein:

```
Format Drive RAD: Name "RAD-Disk"
```

Wir werden nun mit der scheinbar unsinnigen Meldung

```
Insert disk to be initializied in drive RAD:
and press Return
```

dazu aufgefordert, die [Return]-Taste zu drücken. Vielleicht werden Sie darüber lachen, daß der Rechner von Ihnen verlangt, eine Diskette in ein RAM-Laufwerk einzulegen. Dies bringt die offene System-Architktur, nach der der Amiga konzipiert wurde, jedoch mit sich. Ältere Rechner konnten nur wenige verschiedene Speichermedien ansprechen. Zwar konnten auch bei diesen Rechnern neue Gerätetypen angeschlossen werden, dies war aber extrem umständlich. Das Betriebssystem des Amiga, das sich aus mehreren Ebenen zu-

sammensetzt, weiß auf der Ebene des File-Systems gar nicht, auf was es eigentlich zugreift. So kann man durch einen anderen Treiber auch ohne Probleme auf vollkommen andere Medien zugreifen. Verfügt man über einen Disketten-Monitor, der in der Lage ist, beliebige Geräte anzusprechen, kann man RAD: sogar mit ihm bearbeiten.

2.3.13 Wissenswertes über die Tastatur

Die Tastatur des Amiga bietet weit mehr Möglichkeiten, als sich aus der Beschriftung der einzelnen Tastenkappen erkennen läßt. Sinn und Aufgabe dieses Kapitels ist es daher, Ihnen zu zeigen, welche Zeichen und Steuercodes Sie erzeugen können, wenn Sie mit bestimmten Tastenkombinationen arbeiten. Der ASCII-Code ließ ursprünglich nur 128 Zeichen zu, von denen allerdings nicht alle für sichtbare Zeichen, sondern auch einige für Steuercodes verwendet wurden. Der Amiga benutzt einen erweiterten ASCII-Code, der mit 256 Zeichen arbeitet, von denen ebenfalls einige Steuercodes abzuziehen sind.

Steuercodes bewirken z.B. einen Seitenvorschub auf dem Drucker, ein Aufblitzen des Bildschirms und ähnliche Dinge. Bliebe noch anzumerken, daß sich die in diesem Kapitel gemachten Angaben auf die deutsche Tastaturbelegung (d) beziehen. Bei anderen Belegungen können sich Abweichungen ergeben. Da nicht jede Taste ein Zeichen erzeugt, sondern es auch Sondertasten, Mehrfachbelegungen und vieles mehr gibt, werden wir zunächst klären, welche Tastenarten wir unterscheiden müssen.

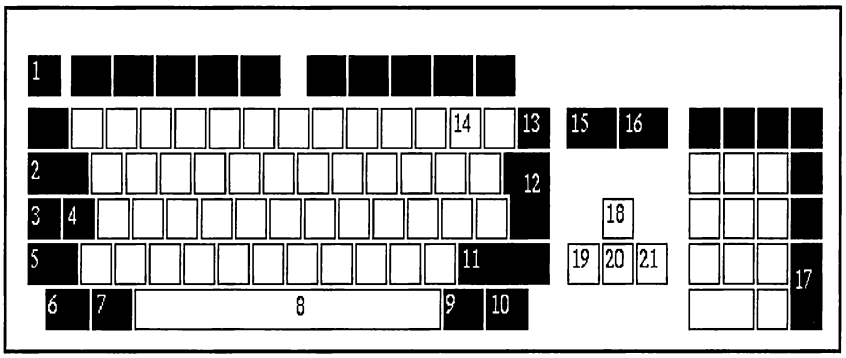


Abb. 2.4b: Tastatur

Normale	Sie erzeugen, wenn sie gedrückt werden, ein Zeichen.
Wahltasten	Sie sind nur in Verbindung mit einer normalen Taste sinnvoll. Drückt man sie, bevor und während man eine normale Zeichentaste betätigt, bewirken sie, daß nicht das reguläre Zeichen dieser Taste ausgegeben wird. Teilweise ist es möglich, mehrere Wahltasten miteinander zu kombinieren, um mehr als zwei mögliche Codes auf eine Taste zu legen.
Vorwahltasten	Die Vorwahltasten haben die gleiche Aufgabe wie die Wahltasten. Sie unterscheiden sich nur in der Art der Anwendung, da sie nicht wie die Wahltasten mit, sondern vor den normalen Tasten betätigt werden.
Sondertasten	Sie geben kein Zeichen aus, sondern übergeben, wenn sie gedrückt werden, einen Code, der für den Rechner eine bestimmte Bedeutung hat, zum Beispiel die [Tab]-Taste.
Zehnerblock	Der Zehnerblock ist ähnlich aufgebaut wie das Tastenfeld einer Registrierkasse. Er ist besonders für die schnelle Eingabe von Zahlen wichtig.

Index-Zahl	Name der Taste
1	[Esc] oder [Escape]
2	[Tab] oder Tabulator
3	[Ctrl] oder Controll
4	[Caps Lock]
5	[Shift]
6	[Alt]
7	Linke [Amiga]-Taste (oder: [Commodore]-Taste)
8	[Space] oder [Leertaste]
9	Rechte [Amiga]-Taste (oder kurz: [Amiga]-Taste)
10	[Alt]
11	[Shift]
12	[Return]
13	[Backspace]
14	[Akzent]
15	[Del] oder [Delete]
16	[Help]
17	[Enter]
18	[↑]
19	[←]
20	[→]
21	[↓]

Einigen Tasten hat man einen Namen gegeben, der sich nicht ohne weiteres sinnvoll erschließen läßt. Für den Computer-Neuling ist es daher unter Umständen ziemlich schwer, z.B. die Tabulator- oder die [Backspace]-Taste zu finden. Um das Problem, das sich aus der Verwendung dieser Begriffe in Zeitschriften und Büchern ergibt, aus der Welt zu schaffen, geben wir Ihnen nun einen Überblick über diese Tasten und ihre Aufgabe. Dabei hilft uns die Abbildung 2.4b, die die Amiga-Tastatur darstellt. Einige Tasten sind auf dieser Abbildung mit Indizes versehen. Auf diese Zahlen beziehen wir uns dann im Text.

[Escape]

Diese Taste erfüllt verschiedene Aufgaben, die je nach Programm unterschiedlich sind. Oft wird sie dazu verwendet, einen Programmteil zu verlassen und ins Hauptmenü bzw. die nächst höhere Programmebene zurückzukehren.

Tabulator

Die [Tab]-Taste ist besonders bei Textverarbeitungen und Editoren wichtig. Sie ermöglicht es, an einen bestimmten Punkt in der Zeile oder im Text zu springen. Sie kann aber wie fast alle Tasten auch für andere Zwecke zweckentfremdet werden. Bei Deluxe Paint II schaltet sie z.B. die Farbenbewegung ein.

[Ctrl]

Diese Taste ist beim Amiga nicht ganz so wichtig wie beim PC. Sie ermöglicht es, auf einfache Weise Steuercodes einzugeben. Dabei wird sie als Wahltaste verwendet, d.h. zusammen mit einer normalen Taste gedrückt. Im allgemeinen wird dann vom ASCII-Code der Taste der Wert 64 abgezogen, so daß dieser im Bereich der Steuercodes liegt.

Caps Lock

Diese Taste ist etwas ganz besonderes. Sie ist mit einer Lampe ausgestattet, die aufleuchtet, wenn man die Taste betätigt. Läßt man sie wieder los, leuchtet sie immer noch - so lange, bis Caps Lock zum zweiten Mal gedrückt wird. Wenn Caps aktiv ist, d.h. die Lampe leuchtet, werden alle Buchstaben, die man eingibt, als Großbuchstaben ausgelegt. Im Gegensatz zum festgestellten "Hochsteller" einer Schreibmaschine wirkt sich Caps nur auf Buchstaben aus. Zahlen- und Sonderzeichentasten werden weiterhin normal behandelt. Dies hat den Vorteil, daß sich Texte versal (in Großbuchstaben) schreiben lassen, wobei Zahlen und Sonderzeichen jedoch normal verarbeitet werden.

Shift

Shift gibt es gleich zweimal. Diese Tasten entsprechen dem "Hochsteller" einer Schreibmaschine. Drückt man eine Taste zusammen mit Shift, wird der Buchstabe groß geschrieben. Bei Tasten, die keinen Buchstaben darstellen, wird, wenn Shift benutzt wird, das Zeichen ausgegeben, das sich in der oberen Hälfte der Tastenkappe befindet.

Alt

Auch die Alt-Tasten sind wie Shift Wahltasten. Mit ihnen werden jedoch eher exotische Zeichen an den Tag befördert, die in anderen Ländern benötigt werden. Im Gegensatz zu vielen anderen Rechnern kann der Amiga-Zeichensatz nämlich in fast allen Ländern verwendet werden, da er alle landesspezifischen Zeichen enthält. Aber auch einige Hilfszeichen können mit Alt erzeugt werden: Brüche, Copyright-Zeichen und vieles mehr. Benutzt man Alt und Shift gleichzeitig, lassen sich noch mehr Zeichen hervorzaubern.

Linke Amiga-Taste

Diese Taste wird sehr selten verwendet. Anders als bei den meisten anderen Tasten kann man mit ihr keine Zeichen erzeugen. Drückt man sie jedoch mit

einer anderen Taste zusammen, kann man dem Bestriebssystem (oder einem anderen Programm) eine Mini-Nachricht übermitteln, ohne ein Eingabe-Fenster geöffnet zu haben. Zur Zeit sind jedoch nur die Buchstaben [N] und [M] mit sinnvollen Funktionen belegt. Werden mehrere Screens dargestellt, befördert [Amiga] [N] die Workbench-Screen nach vorne. [Amiga] [M] bringt sie wieder in den Hintergrund. Zu beachten ist, daß dies nur mit der linken [Amiga]-Taste möglich ist. Im Gegensatz zu [Shift] und [Alt] besteht nämlich ein Unterschied zwischen linker und rechter [Amiga]-Taste. Übrigens kann es sein, daß diverse Programme die linke [Amiga]-Taste auch als Wahltaste verwenden. Uns ist jedoch kein solches Programm bekannt.

Rechte [Amiga]-Taste

Weit wichtiger als die linke ist die rechte [Amiga]-Taste. Viele Programme, die mit Menüs am oberen Rand der Screen arbeiten, gestatten es nämlich, anstelle des gewohnten Verfahrens (rechte Maustaste drücken, Menü runterklappen, Menüpunkt wählen) auch einen Menüpunkt direkt zu wählen. Damit sie nicht immer im Handbuch des betreffenden Programms nachschlagen müssen, zeigt Intuition, also das grafische Betriebssystem, bei jedem Menüpunkt, der über eine Tastenkombination zu wählen ist, eine symbolisierte [Amiga]-Taste und ein Zeichen an. Dieses Zeichen muß zusammen mit der rechten [Amiga]-Taste gedrückt werden, wenn die Menüfunktion ohne Maus aufgerufen werden soll.

[Leertaste] *bzw.* [Space]

Zu dieser Taste gibt es nicht viel zu sagen. Wichtig ist nur, daß Sie ihren Namen kennen. Betätigen Sie sie, wird eine Leerstelle ausgegeben.

[Return] *und* [Enter]

Diese Tasten sind, wenn sie nicht mit einer Wahltaste verbunden werden, gleichwertig. Sie schließen eine Eingabe ab, bringen den Cursor an den Anfang der nächsten Zeile usw.

[Backspace]

Diese Taste hat die Aufgabe, das Zeichen links neben dem Cursor zu löschen. Sie kann natürlich auch anders eingesetzt werden. Das ist allerdings unüblich.

[Akzent]

Bei dieser Taste handelt es sich um die einzige reine Vorwahltaste der deutschen Tastaturbelegung. Drückt man sie, wird das nachfolgend gedrückte Zeichen mit einem Akzent versehen. Diese Taste kann auch mit [Shift] kombiniert werden. Der Akzent zeigt dann in die andere Richtung. Nicht alle Zeichen können mit einem Akzent-Zeichen versehen werden. Betätigt man die [Akzent]-Taste zuerst ohne und dann mit [Shift], werden beide Akzente miteinander kombiniert.

[Delete]

Mit [Delete] soll das Zeichen an Cursorposition gelöscht werden. An dieser Stelle sollten wir vielleicht noch darauf hinweisen, daß der Cursor das Quadrat oder das Zeichen ist, das anzeigt, an welcher Stelle die nächste Eingabe in einer Textverarbeitung, im CLI oder ähnlichen Programmen eingefügt wird.

[Help]

Normalerweise sollte man in einem Programm mittels der [Help]-Taste Hilfen anfordern können, wenn man nicht mehr weiter weiß. Leider ist dies recht selten der Fall. Viele Programme verwenden diese Taste daher zu anderen Zwecken. Textomat setzt sie z.B. dazu ein, den aktuellen Absatz neu zu formatieren.

[↑], [↓], [←] *und* [→]

Die Aufgabe dieser Tasten ist nicht schwer zu erraten. Mit ihnen kann man den Cursor in Programmen steuern, die einen solchen verwenden.

Reset über die Tastatur

Wenn der Rechner abgestürzt ist oder ein Programm nicht beendet werden kann, kann man zwei Dinge tun: den Rechner ausschalten oder einen Reset über die Tastatur auslösen. Dazu müssen wir lediglich drei Tasten gleichzeitig drücken: Ctrl und beide Amiga-Tasten. Doch Vorsicht: Diese Reset-Technik ist nicht dazu geeignet, Viren, die sich aktiv im Speicher befinden, zu entschärfen. Auch kann es Programme geben, die mit einem Reset-Schutz ausgestattet sind, und sich daher nicht über den Tastatur-Reset abbrechen lassen können.

Die Tastatur und das Multitasking

Bei den Rechnern der früheren Generation war es klar, daß nur ein Programm abgearbeitet werden konnte. Da der Amiga dank seines Multitaskings in der Lage ist, mehrere Aufgaben gleichzeitig zu bewältigen, kann es sein, daß mehrere Programme die Tastatur als Eingabegerät verwenden wollen. Zwar kann theoretisch jedes Programm zu jeder Zeit die Tastatur abfragen - in der Praxis sieht es jedoch etwas anders aus. Arbeitet man mit Fenstern und Screens, bekommt immer das Programm die Tastatureingaben, dessen Fenster aktiv ist. Benutzt ein Programm mehrere Fenster, können Eingaben auch in unterschiedlichen Fenstern gemacht werden, wobei sie dann auch unterschiedlichen Verwendungszwecken zugeführt werden können. Doch wann ist ein Fenster aktiv? Das Fenster, in das zuletzt mit der linken Maustaste hineingeklickt wurde, ist meistens aktiv. Doch auch optisch kann man erkennen, welches Fenster die Eingaben bekommt. Es gibt nämlich immer nur ein Fenster, dessen Kopfleiste, in der der Fenstername steht, nicht geriffelt ist - und dieses Fenster ist das zur Zeit aktive. Das aktive Fenster beansprucht jedoch nicht nur die Tastatur, sondern auch die Menüleiste für sich.

Den Zeichensatz voll nutzen

Neben den gewöhnlichen Zeichen kann der Amiga noch einige Sonderzeichen darstellen. Diese werden bevorzugt über Alt-Tastenkombinationen erzeugt. Doch auch einige wichtige normale Zeichen können nur auf diese Weise eingegeben werden.

Zeichen	Tasten-Kombination
Apostroph	Alt Ä
Ausrufungszeichen auf dem Kopf	Alt I
Fragezeichen auf dem Kopf	Alt Shift M
Pfund-Zeichen	Alt L
Doppelstrich	Alt Shift I
Hoher Kreis	Alt W
Plus-Minus	Alt Y
Mittelpunkt	Alt 8 (Haupttastatur)
Hochstrich	Alt Shift N
Breiter Strich	Alt N
Klammeraffe	Alt 2 (Haupttastatur)
Durchstreich X; Mal-Zeichen	Alt X
Geteilt-Zeichen	Alt Shift X
Pi	Alt P
Hoch 1	Alt 1 (Haupttastatur)
Hoch 2	Alt Shift 2 (Haupttastatur)
Hoch 3	Alt 3 (Haupttastatur)
1/4	Alt 5 (Haupttastatur)
1/2	Alt 6 (Haupttastatur)
3/4	Alt 7 (Haupttastatur)
(C)	Alt E
(R)	Alt R
Ein Anführungszeichen unten	Alt M
Ein Anführungszeichen oben	Erst Akzent dann Space
Doppelpfeil nach links	Alt 9 (Haupttastatur)
Doppelpfeil nach rechts	Alt 0 (Haupttastatur)

Neben der Akzent-Taste gibt es noch einige andere Vorwahlmöglichkeiten, die jedoch nicht über eine eigene Taste, sondern über Alt-Kombinationen eingestellt werden.

Gewünschte Ergänzung	Tasten-Kombination
Akzent 1	Alt F
Akzent 2	Alt G
Akzent 1+2	Alt H
Wellenlinie	Alt J
Punkte	Alt K

Ein kleines Beispiel soll uns die obige Tabelle verständlich machen. Nehmen wir einmal an, wir wollen ein großes E mit zwei Punkten darüber erzeugen. Dieses Zeichen wird in einigen Sprachen verwendet. Wir drücken daher also

zunächst [Alt] [K] und dann [Shift] [E]. Wenn wir mit einem Programm arbeiten, das die Tastatur richtig abfragt, erscheint das gewünschte Zeichen. Es ist also nicht mehr schwer, Texte in beliebigen Sprachen (außer Japanisch, Russisch, Chinesich...) zu verfassen. Einige besonders ausgefallene Zeichen können über spezielle [Alt]-Kombinationen direkt erzeugt werden. [Alt]-[Shift]-[A] ist z.B. für ein Zeichen zuständig, das A und E gleichzeitig ist.

Ärger mit der amerikanischen Tastaturbelegung

"Was interessiert mich die amerikanische Belegung der Tastatur?" werden Sie jetzt vielleicht fragen. "Oft kommt es vor, daß eine fremde Boot-Diskette oder ein Programm nur mit der amerikanischen Standard-Belegung arbeitet!" ist die Antwort. Muß man mit einem solchen Programm arbeiten, ist es wichtig, zu wissen, wie man bestimmte Zeichen eingeben kann. Die folgende Tabelle soll Ihnen daher helfen, schnell mit diesem Problem fertig zu werden, ohne für ein bestimmtes Zeichen alle Tasten ausprobieren zu müssen.

Gewünschtes Zeichen	Das muß ich eingeben
y	[Z]
z	[Y]
ö	[Alt] [K] und [O]
ä	[Alt] [K] und [A]
ü	[Alt] [K] und [U]
ß	[Alt] [S]
Y	[Shift] [Z]
Z	[Shift] [Y]
Ö	[Alt] [K] und [Shift] [O]
Ä	[Alt] [K] und [Shift] [A]
Ü	[Alt] [K] und [Shift] [U]
"	[Shift] [Ä]
	[Alt] [Shift] [S]
&	[Shift] [7]
/	[-]
(	[Shift] [9]
)	[Shift] [0]
=	[Akzent]
?	[Shift] [-]
+	[Shift] [Akzent]
*	[Shift] [8]
#	[Shift] [3]
^	[Shift] [6]

Gewünschtes Zeichen	Das muß ich eingeben
;	Ö
:	Shift Ö
-	ß
_	Shift ß
<	< oder Shift ,
>	> oder Shift .
@	Shift 2
[	Ü
]	+
{	Shift Ü
}	Shift +

PC-Belegung des 10er-Blocks

Auf den Tastenkappen des 10er-Blocks befinden sich auf der Frontseite Zeichen und Texte, die scheinbar keine Funktion haben, wie man durch Ausprobieren festzustellen glaubt. In Verbindung mit einigen Programmen, der PC-Emulation und der PC-Karte können sie jedoch genutzt werden. Besonders bei Editoren und Textverarbeitungsprogrammen sind die neuen Belegungen nützlich.

Auf den Tasten 8, 4, 6 und 2 liegen nochmals die Cursorfunktionen. Warum? Weil der erste PC noch keine richtigen Cursortasten hatte und der Amiga ja in der Lage sein soll, auch einen solchen PC zu emulieren. Interessanter sind 7, 9, 1 und 3. Auf diesen Tasten liegen nämlich Funktionen, die dem Amiga normalerweise fremd sind. Mit Home kann man, je nach verwendetem Programm, an den Text-, Seiten- oder Zeilenanfang springen. End bewirkt genau das Gegenteil, d.h. der Cursor wird an das Ende der Zeile, der Bildschirmseite oder des Textes gebracht. PgUp steht stellvertretend für Page Up, also Seite hoch. Bei vielen Programmen kann man mit dieser Taste eine Bildschirmseite nach oben blättern. Auch hierzu gibt es wieder ein Gegenstück: PgDn. Die Page-Down-Taste scrollt eine Seite nach unten. Auf der Punkt-Taste liegt aus Gründen der Kompatibilität und der Gewöhnung nochmals Del. Neu hinzugekommen ist Ins. Die Insert-Funktion fügt eine Leerstelle ein oder schaltet in den Einfügemodus.

Viele Programme gestatten es, die eben genannten Tasten noch mit Shift, Alt und Ctrl zu kombinieren. Bei Textomat, BECKERtext und Profimat

kann man wählen, ob der Zehnerblock normal oder mit PC-Belegung betrieben werden soll. Ich kann Ihnen nur dringend raten, die PC-Belegung zu verwenden!

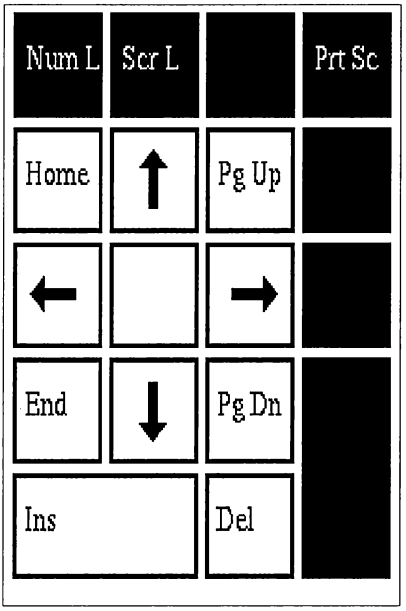


Abb. 2.5: Zehnerblock

Bleiben noch [NumL], [ScrL] und [PtSc]. Mir ist kein Amiga-Programm bekannt, das diese Tasten mit den Funktionen belegt, für die sie eigentlich vorgesehen sind. Die nachfolgende Beschreibung ist daher nur für die PC-Emulation und MS-DOS-Programme interessant.

Mit NumL kann man wählen, ob der 10er-Block mit Zahlen oder Cursorfunktionen belegt werden soll. [ScrL] soll die Bildschirmausgabe stoppen. Einige Programme machen von dieser Möglichkeit Gebrauch, viele lassen es aber auch bleiben. Besonders wichtig ist eine solche Funktion in den meisten Fällen ohnehin nicht. [PrtSc] soll schließlich den Bildschirminhalt ausdrucken (Hardcopy). Um zu verhindern, daß diese Funktion durch ein Versehen ausgelöst wird, muß die Taste zusammen mit [Shift] gedrückt werden.

3. Amiga-Software bringt Leben

Nach einigen Tips zur allgemeinen Arbeit mit dem Amiga stellen wir uns nun die Frage, wie es weiter geht. Für den Einsteiger bietet sich hier als einfachster Weg die Nutzung der vorhandenen Hard- und Software. Wir möchten in den folgenden Kapiteln auf verschiedene Vertreter eingehen und diese für Sie unter die Lupe nehmen. In diese Tests haben wir unsere ganze Erfahrung einfließen lassen, damit es Ihnen leichter fällt, die Auswahl selbst zu treffen.

Software ist das Lebenselexier für unseren Amiga. Ohne geht es nicht! Ob Spiel, Spaß oder Anwendung - man kommt ohne Software nicht aus. Wir betrachten hier natürlich die Anwender-Software, da der übrige Software-Markt zu groß und zu vielschichtig ist. Außerdem wollen wir uns mehr den ernsthaften Programmen zuwenden, bei denen sich ein Preis/Leistungs-Verhältnis objektiv ablesen läßt, was bei Spielen nicht möglich ist.

Software stellt den wichtigsten und zugleich billigsten Weg dar, den Amiga sinnvoll zu nutzen. Es werden derzeit Programme aller Anwendungsgebiete angeboten. Dies bietet dem Anwender und Käufer eine schier unendliche Auswahl, erschwert ihm aber zugleich die Wahl.

Wir haben es uns deshalb zur Aufgabe gemacht, die vier wichtigsten Themen der Anwender-Software herauszupicken und jeweils etwa zwei Programme ausführlich zu besprechen. Unter Abwägung der Möglichkeiten und Interessen, die ein 500er-User hat, haben wir unsere Erfahrungsberichte gestaltet und hoffen so, eine möglichst große Hilfe für Sie geschaffen zu haben.

Dieses Kapitel behandelt Text- und Datenverarbeitungen, die sich auch mit einem Amiga 500 ohne irgendwelche Erweiterungen betreiben lassen. Es bietet weiterhin Einsicht in die Welt der Grafik und der Töne. Alle Gebiete werden mit Beispielprogrammen erklärt und in praktischen Anwendungen dokumentiert.

Damit sollte es Ihnen am Ende möglich sein, selbst abzuwägen, welches Programm für Sie die beste Lösung darstellt.

3.1 Kleine Textverarbeitungen

Bevor wir uns einige ausgesuchte Textverarbeitungen ansehen, sollten wir zuerst abchecken, welche Anforderungen man an solch ein Programm stellen sollte. Wir erarbeiten uns also quasi zuerst die Forderungen und können so nachher vergleichen, inwieweit sie vom Programm erfüllt werden. Sicherlich läßt sich abschließend nicht eindeutig sagen, welches Programm im Einzelfall günstiger ist, aber Sie werden dadurch eine Entscheidungshilfe haben.

Welche Fähigkeiten sollte unsere Textverarbeitung haben?

- Die Eingabe des Textes sollte schnell ablaufen und gleichzeitig auf dem Bildschirm formatiert dargestellt werden.

- Alle Textattribute wie unterstrichen, kursiv und fett sollten vom Programm unterstützt werden und auch gleichzeitig sichtbar im Text dargestellt sein.

- Das Programm sollte menü-gesteuert sein, aber auch die Bedienung über die Tastatur oder ein anderes Medium (wie z.B. die Maus) unterstützen.

- Der Ausdruck sollte einfach, schnell und komfortabel geregelt werden. Dafür müssen einerseits die Einstellungen der Preferences beachtet werden, andererseits muß das Programm aber auch selbständig wichtige Parameter des Druckers einstellen können, um einen zu großen Aufwand zu vermeiden.

- Die Gestaltung des Textes in Hinblick auf das Blatt Papier, auf dem er später erscheinen soll, muß sehr eng an den Ausdruck angelegt sein, damit der Benutzer schon auf dem Bildschirm sieht, was später gedruckt wird.

Damit haben wir schon die wichtigsten Kriterien für eine Textverarbeitung zusammen. Sicherlich stellt jeder seine persönlichen Anforderungen an ein Produkt, doch zu den speziellen Einzelheiten kommen wir dann noch im Verlauf der Tests.

Als Testkandidaten haben wir zwei verschiedene Textverarbeitungen für den Amiga herausgesucht. Zum einen ist dies TEXTOMAT, ein Produkt aus dem Hause DATA BECKER. Wir haben dieses Programm nicht gewählt, weil es genau wie dieses Buch von DATA BECKER stammt, sondern weil es die preiswerteste semiprofessionelle deutsche Textverarbeitung ist. Ob und wieweit diese Behauptung erfüllt ist, werden wir dann im Testbericht sehen. Der zweite Bewerber ist UBM Text, ein Schreibprogramm der UBM-Drecker GmbH. Auch UBM Text ist einigermaßen preiswert und bietet alle wichtigen Funktionen.

Doch beginnen wir nun mit dem Testbericht:

TEXTOMAT

Beim Kauf von TEXTOMAT erhält man ein 210 Seiten starkes deutsches Handbuch zusammen mit einer 3½-Zoll-Diskette. Auf ihr befindet sich das Programm, das nicht kopiergeschützt ist. Allerdings erhält man eine Seriennummer, die vor Inbetriebnahme der Software eingegeben werden muß. Nur wenn dies ordnungsgemäß gemacht wurde, besteht die Möglichkeit, bei späteren Verbesserungen den Update-Service von DATA BECKER wahrzunehmen. Der Preis von TEXTOMAT liegt mit 99,- DM deutlich unter dem anderer Textverarbeitungen.

Mit Textomat arbeiten

Das Programm selbst präsentiert sich mit einem Window auf der Workbench-Screen. Es kann einerseits über die boot-fähige Diskette gestartet oder aber von der Workbench aufgerufen werden. TEXTOMAT verbraucht wenig Speicher. Dies ist besonders dann ein wichtiges Kriterium, wenn man mit nur 512 KByte arbeitet oder aber parallel noch andere Programme laufen lassen möchte.

Die Texteingabe selbst gestaltet sich unkompliziert. Der Cursor folgt ohne Probleme dem Text. Hierbei fällt auch der automatische Zeilenumbruch auf. TEXTOMAT ergänzt (bzw. füllt) jede Zeile mit Füllpunkten, die den freien Raum deutlich machen. Um eine Zeile möglichst effektiv zu nutzen, werden die Wörter getrennt. Der dafür verwendete Algorithmus arbeitet nach Silben,

was nicht immer zu richtigen Resultaten führt. Allerdings kann man mit einer Tastenkombination in den zu trennenden Worten jederzeit eigene Trennungsvorschläge einfügen, die dann befolgt werden.

Speichern und Laden von Texten

Geht es ans Sichern des Textes, tritt man das erste Mal mit den Funktionen des Programms in Berührung. Jede läßt sich über eines der vier vorhandenen Menüs aufrufen. Das erste Menü enthält die Datei-Funktionen und ist damit genau richtig für das Abspeichern. TEXTOMAT greift nach dem Aufruf dieser Funktion zu zwei Requestern. Der erste fragt nach dem Text-Typ, in dem die Zeilen abgespeichert werden sollen bzw. der gesichert werden soll. Beim Text lassen sich auch Anfangs- und Endzeile bestimmen, so daß man nur Blöcke (Teile des Textes) abspeichern kann. Der zweite Requester ist dann die eigentliche File-Request-Box, in der der File-Name und das Verzeichnis bestimmt werden. Diese Request-Box ist etwas unglücklich programmiert, denn TEXTOMAT lädt bei jedem Entfernen einer Diskette aus irgendeinem Laufwerk das Verzeichnis neu, was nicht einmal dann unterbrochen werden kann, wenn man schon alle Angaben weiß. Ansonsten ist die File-Auswahl aber gut durchdacht.

Die Textgestaltung

Die Textgestaltung macht TEXTOMAT keine Probleme. Als Schriftarten werden alle bekannten Modi, die auch die Drucker unterstützen, im dritten Menü angeboten. Neben fett, kursiv, unterstrichen, Hoch- und Tiefstellen ist die partielle Auswahl von Near-Letter-Quality möglich. Die Zeichendichte (die Anzahl der Zeichen pro Zeile bei gleicher Breite) kann aus vier verschiedenen Möglichkeiten ausgewählt werden, wenn es der Drucker unterstützt. Sogar der Zeilenabstand kann, genau wie bei der herkömmlichen Schreibmaschine auch, von 1, 1½ oder 2 variiert werden. Damit sind alle Attribute, die zur Textgestaltung nötig sind, vorhanden, so daß dem Anwender keine Wünsche mehr offen bleiben sollten. Ein besonders hervorzuhebender Aspekt ist, daß alle Druck-Modi auf dem Bildschirm dargestellt werden. Selbst Kombinationen wie Hochsetzen plus Kursiv- und Fettschrift verarbeitet TEXTOMAT anstandslos.

Sehen Sie hier einen Beispieltext, der alle Schriftattribute in ihren Gestaltungsmöglichkeiten darstellt:

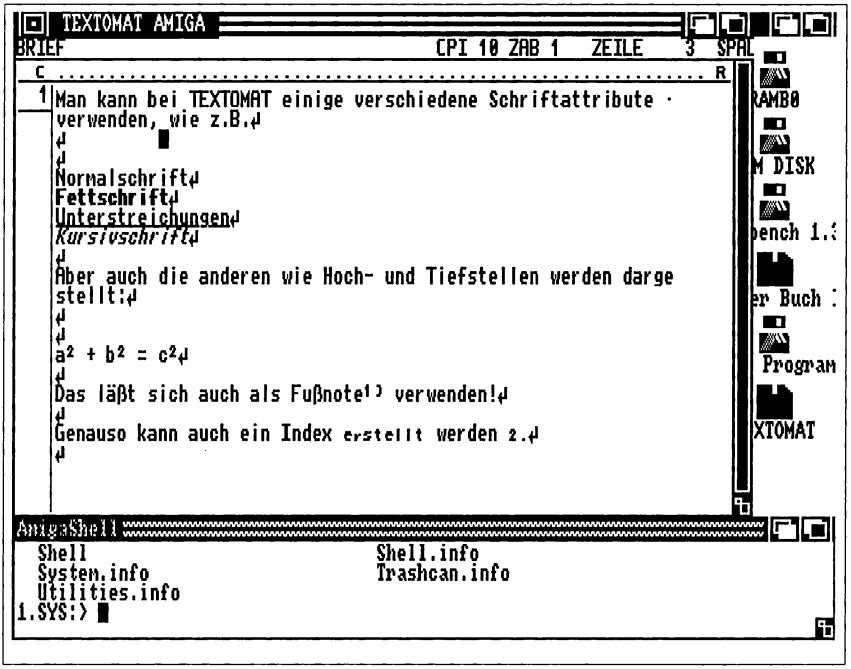


Abb. 3.1: TEXTOMAT-Darstellung der Schriftattribute

Den Text bearbeiten

Neben der Darstellung von Schriftattributen ist aber auch die Textbearbeitung ein wichtiges Kriterium für die Qualität einer Textverarbeitung. Das zweite Menü ist dafür verantwortlich. Es enthält zuerst einmal die beiden Standard-Funktionen Suchen und Ersetzen. Beide werden mit dem gleichen Requester bearbeitet. Dort kann entweder der zu suchende bzw. zu ersetzende Text eingegeben werden, oder aber man übernimmt den ersten Absatz aus dem vorhandenen Text. Diese sehr ungewöhnliche Auswahl wurde gewählt, um auch das Suchen nach Schriftarten bzw. Zeileneinstellungen zu ermöglichen, was durch die einfache Eingabe von maximal einer Zeile innerhalb eines String-

Gadgets natürlich nicht zu machen ist. Allerdings vermißt man hierbei doch das Zurückgreifen auf die Möglichkeit, den markierten Block als Such- bzw. Ersetzkriterium anzunehmen. Damit wäre nämlich die Spitze des Komforts erreicht.

Wo wir gerade beim Stichwort Blöcke sind; das Markieren von Blöcken ist bei TEXTOMAT sehr praktisch gelöst. Der Cursor kann jederzeit mit einem Klick auf die linke Maustaste positioniert werden. Hält man die linke Maustaste niedergedrückt, wird der Cursor zu einem rechteckigen Rahmen, der seinen Anfangspunkt gefunden hat und beliebig vergrößert werden kann. Der eingeschlossene Text ist nach dem Loslassen markiert. So lassen sich leicht Absätze oder Zeilen-Blöcke markieren. Für einzelne Worte genügt ein Doppelklick auf das Wort. Zum Verkürzen oder Verlängern des markierten Bereichs reicht auch hier wieder ein Doppelklick auf die entsprechende Stelle im Text.

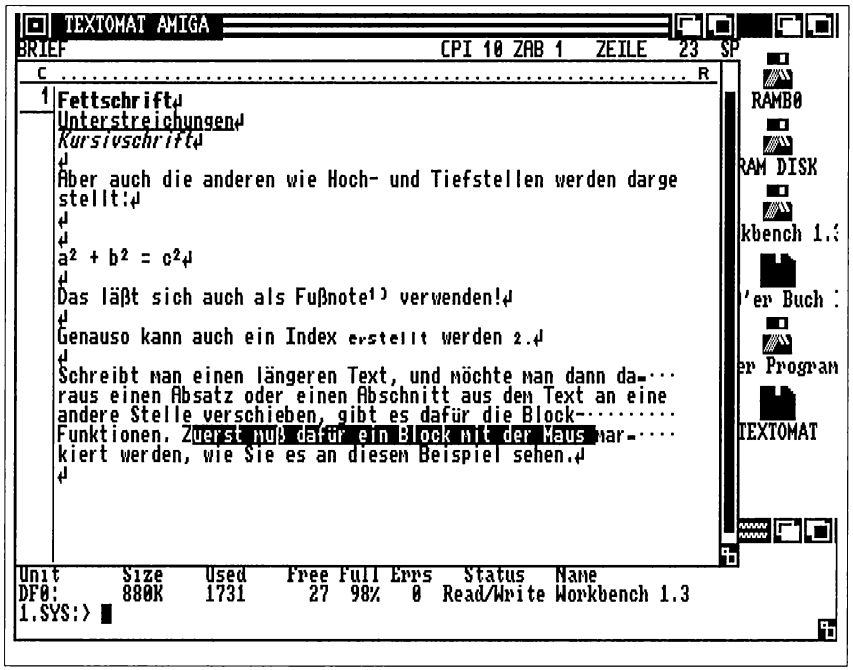


Abb. 3.2: Ein markierter Textblock

Diese so definierten Blöcke können kopiert, gelöscht oder verschoben werden. Alle diese Operationen laufen angenehm schnell, werden aber bei sehr langen Texten entsprechend langsamer.

Zusätzlich und sehr interessant sind im Bearbeiten-Menü die Funktionen Datum und Uhrzeit. Beide tragen den gewählten Wert in den Text ein. Aber damit nicht genug! Dieser Wert ist nicht statisch. Er wird bei einem Ausdruck, beim neuen Laden des Textes oder bei einem Druck auf die Help-Taste aktualisiert. Sie haben so im Briefkopf immer das richtige Datum und können sogar die aktuelle Uhrzeit in einen Text mit einfließen lassen.

Einstellungen am Text

Kommen wir jetzt zu den Einstellungen, die am Text vorgenommen werden können. Unter diesem Stichwort faßt das vierte Menü allgemeine Programm-Einstellungen und die am Text, sprich Formatierung, zusammen.

Sehr angenehm und optisch wirkungsvoll ist das Einrücken. Hiermit lassen sich Absätze oder Passagen im Text um beliebig viele Zeichen einrücken, um sie hervorzuheben oder als Anmerkung zu kennzeichnen. Normalerweise würde man so vorgehen, daß man vor jede Zeile die entsprechende Anzahl von Lücken setzt. Nicht so bei TEXTOMAT. Wird ein Bereich oder Absatz markiert und eingerückt, braucht man sich nicht um die Lücken zu kümmern. Die Zeilen fangen dann einfach später an, und die Arbeit läuft wie gewohnt. Auch nachträglich bietet diese Funktion große Arbeitsersparnis, denn die Breite kann jederzeit neu gesetzt werden.

Wo wir gerade bei der Breite des linken Randes sind, können wir unseren Blick auch auf die Textbreite richten. Sie wird ja im allgemeinen über die Seiten-Parameter eingestellt, zu denen wir gleich kommen. Davon unabhängig ist es aber auch möglich, die Breite ganz individuell pro Zeile oder Absatz zu bestimmen. Sehr interessante Textformatierungen sind so leicht zu bewerkstelligen, weil TEXTOMAT den Bereich immer wieder neu formatiert.

Kommen wir jetzt aber zum Formular: Als Formular bezeichnet das Programm eine Seite des Textes. Hierfür werden einige Einstellungen gebraucht, die in einem großen Requester abgefragt werden. Das sind im einzelnen:

Der wichtigste Wert ist die Papierlänge. Hier gibt man an, wie viele Zeilen pro Seite gedruckt werden können. Als Standard erwartet man hier 72 Zeilen. Wichtig ist dann die davon bedruckbare Höhe. Damit bezeichnet man den Platz, der frei bleibt, wenn man Kopf und Fuß jeder Seite abrechnet. Verwenden Sie einen zweizeiligen Kopf- und Fußtext, haben wir schon vier Zeilen weniger.

Oder aber zwischen Kopf und normalem Text sollen zwei Zeilen frei bleiben. Auch dann sind nur noch 68 Zeilen mit dem Text bedruckbar. Dementsprechend zeigt TEXTOMAT auch beim Fließtext die Seitenumbrüche an, so daß Sie immer wissen, wo eine Seite endet und eine neue beginnt.

Wie selbstverständlich haben wir schon die Begriffe Kopf- und Fußtext verwendet. Doch so selbstverständlich ist das gar nicht! TEXTOMAT bietet für einen Text, der am Anfang bzw. am Ende jeder Seite gedruckt werden soll, jeweils zwei Zeilen, in denen nicht nur einfacher Text, sondern auch Seitennummern, Datum, Uhrzeit und Schriftarten eingebaut werden können. Diese werden allerdings über Steuersequenzen eingeflochten, was nicht sehr komfortabel ist.

Auch die Beschränkung auf nur zwei Zeilen macht es bei Texten schwer, wo drei Zeilen nützlicher gewesen wären. Aber hier geht man den Kompromiß mit der einfachen Eingabe ein.

Kehren wir jetzt zurück zu den Funktionen zum Formatieren des Textes. Dort sind wir nämlich noch längst nicht fertig. Sehr vielseitig bei der Textgestaltung sind die Ausrichtungen. Sie erlauben die Standard-Varianten linksbündig, rechtsbündig, zentriert und Blocksatz. Hiermit läßt sich ein Text optimal gestalten. Linksbündig wird für Flattersatz verwendet, rechtsbündig ist mehr eine Spielerei. Das Zentrieren verwendet man, um Überschriften optimal in die Mitte zu rücken (auch wenn man nachträglich noch etwas verändert, wird es doch immer wieder in die Mitte gerückt). Als letztes steht der Blocksatz auf der Liste, der auch die wichtigste Ausrichtung darstellt. Er wurde erst richtig durch den Computer möglich, denn von Hand war es immer sehr aufwendig, einen Absatz neu mit den entsprechenden Lücken zu versehen, wenn sich die Zeilenlänge änderte.

Sehen Sie jetzt hier zum Abschluß eine Seite, auf der alle Formatierungen noch einmal angewendet wurden:

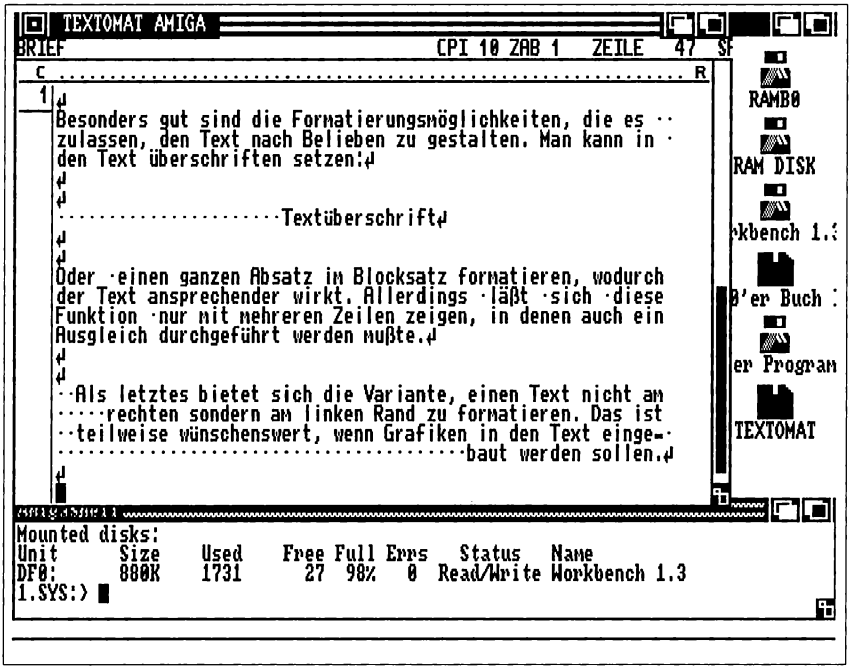


Abb. 3.3: Alle Formatierungs-Möglichkeiten

Besonderheiten

Damit haben wir alle wichtigen Funktionen von TEXTOMAT angesprochen. Von den elementaren Eingabe-Befehlen über die Disketten-Funktionen, weiter über die Text-Gestaltung bis hin zur Formatierung haben wir alle Themen angesprochen. So bleibt es, am Schluß noch auf eventuell vorhandene Besonderheiten hinzuweisen, die nicht unbedingt bei jeder Textverarbeitung zu finden sind.

Weil TEXTOMAT als Textpuffer einen statischen Speicherbereich benutzt, d.h. dieser in seiner Größe nicht verändert werden kann, und man deshalb irgendwann in Speichernot gerät, wird eine Info-Funktion angeboten. Sie gibt Auskunft über die vom Text belegten Prozente des Speicherblocks. Außerdem

findet man hier die persönlichen Daten wieder, die beim ersten Starten des Programms eingegeben wurden.

Zu dem normalen Eingabe-Modus gibt es einen zweiten, der für C-Programme gedacht ist. Wie Sie vielleicht wissen, werden C-Programme durch Einrücken bestimmter Text-Bereiche besonders optisch strukturiert. Dies erfordert sehr viel Arbeit, weswegen viele Autoren darauf verzichten. Mit dem Eingabe-Modus C-Source übernimmt TEXTOMAT diese Arbeit für Sie!

Bei jedem Öffnen einer neuen geschweiften Klammer wird um drei Zeichen weiter eingerückt, und umgekehrt läuft es bei den geschlossenen geschweiften Klammern.

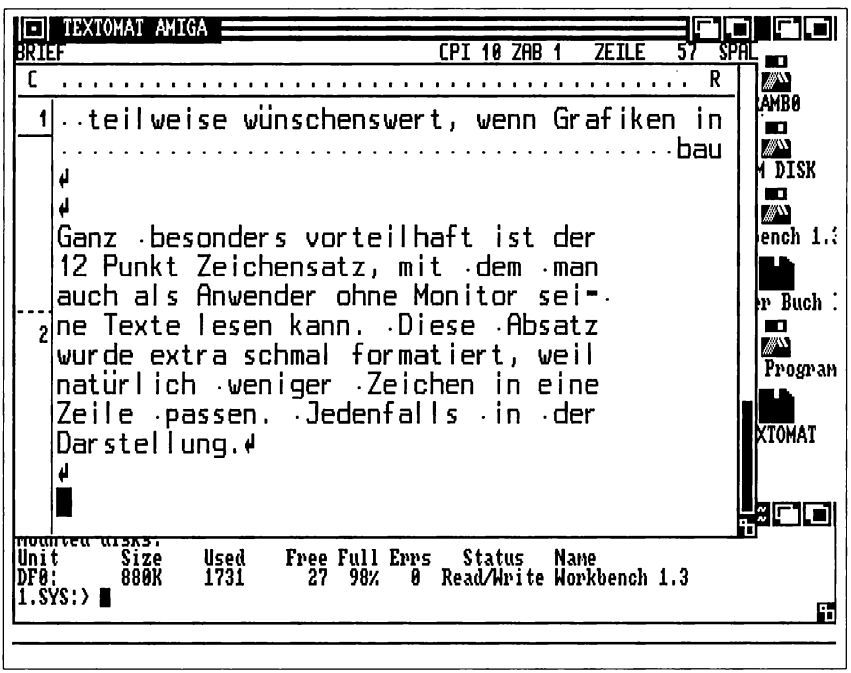


Abb. 3.4: Die Textdarstellung im 12x12-Punkt-Zeichensatz

Wichtig gerade für finanzschwache Amiga-User ist die Möglichkeit, über den Konverter auch einen ganz normalen Fernseher anzuschließen. Dadurch spart man zuerst das Geld für den Monitor, und weil der Amiga sowieso einen 60-

Zeichen-Modus anbietet, ist das kein Problem. Leider benutzt TEXTOMAT aber einen eigenen Zeichensatz, den es nicht als 60- und 80-Zeichen-Version gibt. Heißt das, daß Sie deshalb auf diese Textverarbeitung verzichten müssen? Nein! Es gibt einen besonders großen 12x12-Punkt-Zeichensatz, der wirklich von jedem Fernseher lesbar wiedergegeben wird. Die Abbildung 3.4 zeigt diesen Zeichensatz in voller Aktion.

UBM-Text 2.2

UBM-Text ist eine weitere Textverarbeitung, die wegen ihres deutschen Herstellers neben TEXTOMAT sehr interessant ist. Auch hier sind die ersten Probleme, man könne die deutschen Sonderzeichen nicht verwenden, schon am Anfang verflogen. Genauso bewegt sich der Preis mit 198,- DM in einem annehmbaren Bereich. Da das Programm auch spielend mit nur 512 KByte auskommt, steht nichts mehr einem Test im Wege. Denn mit diesen Voraussetzungen ist jeder Amiga-500-User angesprochen.

Der Lieferumfang

Beim Kauf von UBM-Text erhielten wir genau wie bei dem ersten Produkt ein Anleitungsbuch von mittlerer Stärke, dem eine Diskette beiliegt. Auch diese Diskette ist nicht mit einem Kopierschutz versehen, besitzt dafür aber eine Seriennummer, unter der man sich als Käufer registrieren lassen muß, um Updates oder Ersatz für gelöschte Disketten anfordern zu können.

Zum Handbuch

Gleich zu Anfang möchte ich ein paar Worte zum Handbuch verlieren. Es ist im NLQ-Modus auf einem guten Drucker erstellt und reproduziert worden. Deshalb fallen auch die nur jeweils einseitig bedruckten Seiten auf. Dadurch halbiert sich natürlich noch die Dicke des Handbuchs, ganz abgesehen von dem äußerst starken Papier.

Das Handbuch gliedert sich in 12 Kapitel, in denen alle Funktionen des Programms beschrieben werden. Nach einem Einführungsteil, der den Update-

Service beschreibt, wird auf die Hardware-Voraussetzungen hingewiesen und dann endlich das Programm gestartet. Leider beschreibt das Handbuch nun ziemlich stur nacheinander die einzelnen Funktionen der Menüs und der Auswahlfelder. Es gibt leider keine didaktische Einführung, in der man Schritt für Schritt mit der Bedienung des Programms vertraut gemacht wird und sich langsam zu den komplizierteren Funktionen vortasten kann.

Für diejenigen, die sich aber schon mit Textverarbeitungen auskennen, dürfte es keine Schwierigkeit darstellen, UBM-Text zu bedienen, da es sich an den allgmeinen Standard hält und nicht unangenehm aus dem Rahmen fällt. Auch die Strukturierung der Menüs und Auswahlfelder läßt eigentlich keine Fehlbedienung zu, so daß man das Handbuch mehr als Nachschlagewerk betrachten sollte.

Mit UBM-Text arbeiten

Nach dem Start von UBM-Text, der übrigens von der Workbench oder vom CLI aus geschehen kann, zeigt sich das Programm auf einer eigenen Screen. Das ist einerseits positiv zu betrachten, weil man so nicht mit anderen Programmen auf der Workbench in Konflikt gerät, andererseits verbraucht das aber auch einigen Speicher, den man vielleicht nicht übrig hat.

Wir finden nun einen Screen vor, von der zwei Drittel zur Texteingabe benutzt werden können. Am rechten Rand ist eine Leiste, auf der die wichtigsten Funktionen als Auswahlfelder zu finden sind. Damit erspart man sich den lästigen Weg über die Menüs. Im unteren Rand sind noch vier Einstell-Felder, die für Überschreiben/Einfügen, Umbruch oder Hilfen stehen. Außerdem stehen hier Angaben zur Zeilen- und Seitennummer. Der Eingabebereich selbst hat am Kopf eine Positions-Zeile, an der sich die Cursorposition ablesen läßt und wo wir auch die Tabulatoren finden.

Die Eingabe ist bei UBM-Text nicht so flüssig wie bei TEXTOMAT. An vielen Stellen sind in unserer Version des Programms noch Fehler. So funktioniert das vertikale Scrolling nicht immer korrekt, so daß manchmal der Cursor außerhalb des Eingabe-Bereichs steht oder die Eingabe nicht wieder zurück-scrollt.

Texte laden und speichern

Zum Speichern eines gerade erstellten Textes bedient sich auch UBM-Text der Menüs. Im Text-Menü werden die Funktionen Text Laden und Speichern angeboten. Auch hier wird man nicht übermäßig mit Komfort überschüttet, zumal nicht einmal der Inhalt der Diskette angesehen werden kann und man auch bei den Devices nur zwischen den fest eingestellten Laufwerken DF0, DF1 und JH0 wählen kann. Alles andere ist nicht erlaubt. Wesentlich positiver als bei TEXTOMAT ist aber das Inhaltsverzeichnis beim Laden. Hier holt UBM-Text erst das Verzeichnis, wenn man den Pfad eingegeben hat und dies bestätigt.

Zusätzlich zum eigenen Format bietet UBM-Text noch die Möglichkeit, den Text in ASCII abzuspeichern, was sich besonders zum Austausch mit anderen Textverarbeitungen anbietet. Hier steht es TEXTOMAT in nichts nach. Etwas anders wurde allerdings das Nachladen von Texten gelöst. Während man bei TEXTOMAT jederzeit einen beliebigen Text dazuladen kann, setzt man bei UBM auf die separate Abspeicherung von Text-Blöcken, die nachgeladen werden können. So baut man das gesamte System strukturierter auf, verlangt allerdings auch eine sehr aktive Mitarbeit vom Benutzer. Gut dabei gelöst ist, daß UBM-Text automatisch für jede Textart ein Standard-Verzeichnis benutzt. Man kann sich natürlich immer noch anders entscheiden. Aber die Ordnung auf der Diskette wird angestrebt und unterstützt.

Den Text bearbeiten

Neben den bekannten Funktionen zur Eingabe des Textes wie der [Return]-Taste oder der Cursortasten besitzen die Textverarbeitungen noch einige Hilfsfunktionen, mit denen man Blöcke und andere Textbausteine bearbeiten kann. Man kann sehr komfortabel Blöcke markieren, indem man sich in den entsprechenden Modus begibt, den Textblock mit dem Cursor abfährt und dann den Modus wieder ausschaltet. Der Bereich wird ab sofort farblich dargestellt und ist sofort erkennbar. Alle weiteren Funktionen beziehen sich nun darauf.

Der Textbaustein kann zuerst einmal gelöscht, kopiert oder verschoben werden. Das ist hilfreich für die Umgestaltung des Textes und zur nachträglichen Gliederung. Außerdem kann dieser Bereich mit den bekannten Schriftattribu-

ten fett, kursiv und unterstrichen versehen werden. Zusätzlich ist noch hoch- und tiefgestellte Schrift möglich, die aber nicht in jedem Fall auf dem Bildschirm dargestellt wird.

Man muß sich dazu auf dem Hilfsbildschirm befinden, bei dem auf der rechten Seite noch die wichtigsten Funktionen aufgeführt sind. Sie sehen dies in der folgenden Grafik:

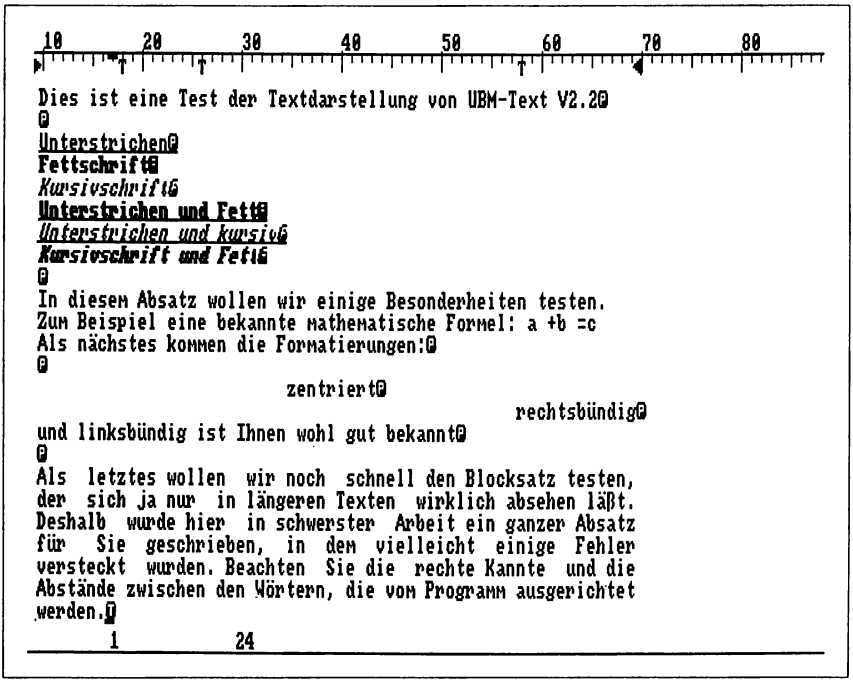


Abb. 3.5: UBM-Text und die Textdarstellung

Wie Sie sehen, werden alle Schriftattribute deutlich dargestellt. Auch Zentrieren, rechts- und linksbündiges Ausrichten sind implementiert, zumal diese Einstellungen besonders häufig gebraucht werden (siehe Überschriften oder das betont gute Aussehen eines Textes).

Beim Bearbeiten von Texten wird man angenehm überrascht! Auf das einfache Markieren eines Blockes haben wir schon hingewiesen. Ebenso einfach ist die Suchfunktion gestaltet. Nach der Anwahl über die Tastatur oder ein Aus-

wahlfeld erscheint ein großer Requester, in dem der Suchtext eingegeben wird. Dazu sind noch einige Einstellungen zu machen. Z.B. kann man die Groß- und Kleinschreibung bei der Überprüfung außer Acht lassen und auch die Suchrichtung angeben. Gerade diese beiden Einstellungen erleichtern oft die Arbeit. Leider sind beide nicht bei TEXTOMAT zu finden. Bei der Such- und Ersetz-Funktion geht es mit gleichem Komfort zu. Hinzu kommen der Text, der ersetzt werden soll, und eine Einstellung, die festlegt, ob bei jedem Fall noch einmal nachgefragt werden soll. Besonders positiv fällt die Möglichkeit auf, zwischen dem Such- und Ersetz-Begriff einen Austausch vorzunehmen, um so den Vorgang schnell wieder rückgängig machen zu können.

Besonderheiten

Genau wie TEXTOMAT kann UBM-Text natürlich auch mit einigen Besonderheiten aufwarten. Als außergewöhnlich kann der Umstand gewertet werden, daß über ein Menü mit Namen Tastatur die Belegung direkt im Programm gewählt werden kann. Schreibt man in einer anderen Sprache, wie z.B. Englisch, Französisch oder Italienisch, ist es kein Problem, schnell auf die richtige Tastaturbelegung umzustellen - und sofort sind alle Sonderzeichen ohne Umstände erreichbar.

Beim Ausdruck geht UBM-Text ebenfalls einen anderen Weg. Wie es von den Entwicklern des Amiga vorgesehen ist, wird im Normalfall die Ausgabe direkt an die Schnittstelle PRT: gesendet. Wer will, kann aber auch auf PAR: ausweichen, weil sich so einige Steuerzeichen übertragen lassen, die PRT: nicht zuläßt. Über Sonderzeichen ist es möglich, Steuerzeichen in den Text einzubauen.

3.2 Datenverwaltungen verschiedener Art

Die Datenverwaltung ist neben der Textverarbeitung eine der wichtigsten Anwendungen auf dem Computer überhaupt. Das, was hier im kleinen Stil auf dem Amiga betrieben wird, machen große Firmen in ihren Rechenanlagen Tag für Tag. Und genauso läßt sich der Vorgang auch von der anderen Seite betrachten. Das, was große Firmen Tag für Tag mit dem Computer erledigen, können wir jetzt auch im Kleinen auf unserem Amiga.

Eine Datenverwaltung ist ein Programm, das die ihm gegebenen Daten in strukturierter Form aufbewahrt, so daß man

1. die Daten sehr schnell in der eingegebenen Form abrufen kann,
2. die Daten immer sortiert vorliegen hat, damit eine schnellere Suche möglich ist und
3. die Daten verknüpfen kann, um so neue Beziehungen herzustellen.

Diese drei Punkte sind die bedeutenden Kennzeichen einer Datenverwaltung. Die erste Anforderung ist einfach nur die, die auch den Karteikasten, der von der Datenverwaltung ersetzt wurde, kennzeichnete. Eine Datenverwaltung muß ihre Arbeit mindestens wie ein konventioneller Karteikasten erfüllen. Das ist selbstverständlich. Mit der zweiten steigen wir erst in das Computer-Zeitalter ein, denn die Fähigkeit, die Daten immer sortiert aufzubewahren, ist sehr arbeitsintensiv. Aber gerade dafür nimmt man ja seinen Computer.

Was den dritten Punkt belangt, so stellt er eine Besonderheit dar, die leider nicht jede Datenverwaltung anbietet. Doch gerade hier zeigt sich, ob das Programm leistungsfähig ist oder nicht.

DATAMAT

Beginnen wir nun mit dem Test der ersten Datenverwaltung. Auch hier haben wir wieder ein Produkt aus dem Hause DATA BECKER. Es ist DATAMAT, eine Datenverwaltung, die grafik-orientiert arbeitet. Das heißt für den Benutzer, daß sämtliche Arbeiten über die Bedienelemente Intuitions erledigt werden und nur wenig Texteingabe nötig ist. Diese Methode bietet sich auch an, denn schließlich ist der Amiga ein Computer, bei dem die Bedienung über die Bildschirm-Grafik im Vordergrund steht.

Der Lieferumfang

Beim Kauf von DATAMAT für 99,- DM erhält man ein etwa 250 Seiten starkes Handbuch mit einer Diskette im Plastikschuber. Die Diskette ist nicht mit einem Kopierschutz, dafür aber mit einer Seriennummer versehen, die wie bei TEXTOMAT vor dem ersten Start des Programms installiert werden muß.

Das Handbuch enthält neben der Bedienungsanleitung am Anfang drei Einsteiger-Kapitel, in denen das Starten von DATAMAT, die Bedienung der Workbench und das Erstellen der ersten Datei für den Anfänger erklärt werden.

Die anderen sieben Kapitel gehen in Abschnitten auf jedes der fünf Menüs ein und behandeln weiterhin das Vorgehen zum Einrichten einer neuen Datei und die Funktionen des Masken-Editors.

Mit DATAMAT arbeiten

Zum Starten der Dateiverwaltung werden einem drei Wege zur Verfügung gestellt. Der erste über das Booten der Workbench-Diskette vom DATAMAT ist sicherlich der einfachste. Aber auch von der Workbench läßt sich die Datenverwaltung aufrufen, wenn man erst nachträglich Daten sichten muß.

Natürlich bleibt als letztes noch der Weg über das CLI, der aber in den meisten Fällen nur ungern gegangen wird.

DATAMAT kommt in seiner jetzigen Version mit 512 KByte aus. Allerdings schaltet das Programm dann zwei Farben von der Workbench-Screen aus, denn es öffnet eine eigene Screen, die zusätzlichen Speicher verbraucht.

Empfehlenswert sind für das extensive Arbeiten mit der Datenverwaltung 1 MByte, da man so auch die Möglichkeit hat, nebenbei andere Programme laufen zu lassen. Aber auch in der Grundversion des Amiga 500 können alle Funktionen des Programms ausnahmslos genutzt werden.

Gleich zu Beginn des Programms ruft dieses automatisch die Funktion zum Öffnen einer neuen Datei auf. Da man dies sowieso tun muß, erspart man sich gleich einen Arbeitsschritt.

Nachdem man hier den Dateinamen eingegeben hat (wir wollen die Datei Hilfe auswählen, die sich schon auf der Diskette befindet, und mit der sich einige Experimente machen lassen) wird die Datei geladen und dafür ein neues Fenster geöffnet. Mit diesem ersten Fenster zeigt sich schon die Stärke von DATAMAT:

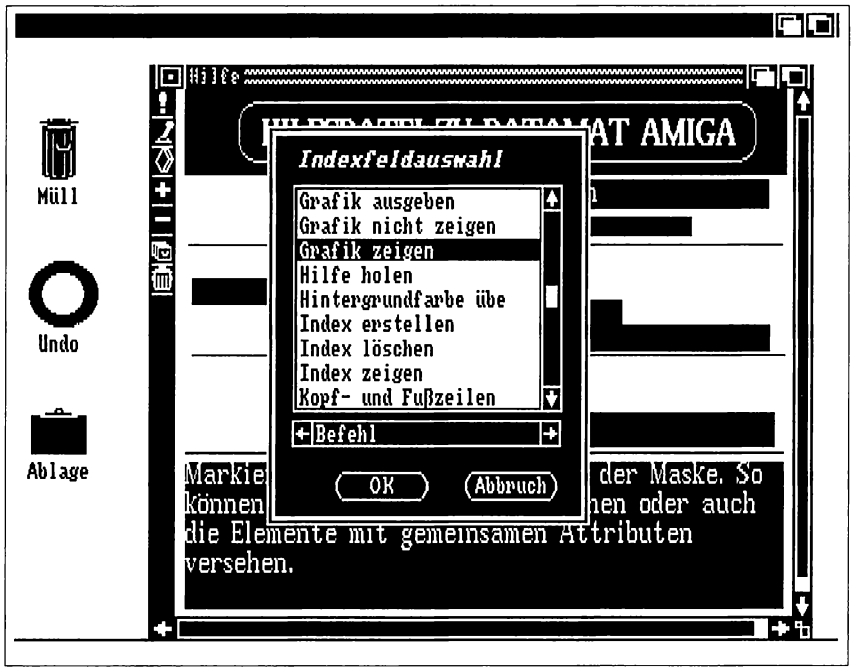


Abb. 3.6: DATAMAT-Arbeitsfläche mit einer geöffneten Datei

Die Bedieneroberfläche

Jedes auf der DATAMAT-Screen geöffnete Fenster steht für eine separate Datei. Es besitzt auf der linken Seite eine Reihe von Symbolen, mit denen sich schnell die wichtigsten Funktionen aufrufen lassen. Innerhalb des Windows sind alle Eingabefelder, und in der Titelleiste der Screen finden wir - ist das Window aktiv - eine Kurzübersicht über die Anzahl der Datensätze, den aktuellen Index und die angewählte Betriebsart.

Sehen wir uns zuerst die einzelnen Symbole an, die im Window-Rand untergebracht sind. Als erstes finden wir ganz oben ein Ausrufezeichen. Es kennzeichnet den Modus der Dateneingabe. Durch Klicken auf das darunter liegende Feld, einen Schalter, kann ohne Probleme der Suchmodus eingeschaltet werden, der mit einem Fragezeichen identifiziert wird.

Das Radiergummi macht kurzen Prozeß mit Daten, die zwar schon in die Maske eingegeben wurden, aber noch nicht abgespeichert worden sind. Nach einem kurzen Klick auf das Zeichen sind alle Felder der Maske gelöscht. Dies wird zum Vorbereiten der Eingabe empfohlen, wenn man vorher im Suchmodus Daten betrachtet hat.

Um sich einmal ganz schnell die Datei anzusehen, blättert man im alltäglichen Leben einfach die Karteikarten durch. Dies gestattet DATAMAT mit den beiden Zeichen Plus und Minus, die jeweils den nächsten oder vorhergehenden anzeigen. Befindet man sich schon an einem der beiden Enden, erscheint eine Fehlermeldung in einem Fenster, die man durch eine Bestätigung entfernen kann. Diese Bestätigung fällt aber manchmal sehr lästig auf, wenn man sich gerade an den Enden einer Datei hin- und herbewegt. Hier wäre es sinnvoller gewesen, sie mit einem gewissen Zeittakt zu versehen oder die Nachricht in die Titelzeile zu schreiben.

Ist die Datei nach einem der Datenfelder geordnet, kann man diesen Wert als Suchbegriff verwenden, den sogenannten Index. Für eine schnelle Betrachtung und das damit verbundene schnelle Anspringen gibt es wiederum ein Symbol. Der Karteikasten öffnet nach dem Anklicken ein Fenster, in dem alle Inhalte dieser Felder der gesamten Datei sortiert aufgelistet sind. Sie müssen nur noch, um sich ein Feld anzusehen, den Begriff anklicken und gelangen mit OK in den Datensatz.

Die letzte der Schnell-Anwahl-Funktionen ist das Löschen des gesamten Datensatzes. Damit wird die Karteikarte - bildlich gesprochen - aus dem Kasten entfernt und weggeworfen.

Alle weiteren Funktionen des Programms können über die Menüs aufgerufen werden wie auch diese, die im Window als Gadgets angeboten werden. Dadurch kann man über die ShortCuts der Menüs auf Tastaturbedienung umsteigen, wenn der ständige Wechsel zwischen Maus und Tastatur lästig wird.

Eine eigene Datei erstellen

Funktionsweise und Aufbau einer Dateiverwaltung lassen sich am besten demonstrieren, wenn eine neue Datei erstellt wird. Denn bei der Erstellung einer benutzbaren Datei stößt man auf fast alle Funktionen des Programms. Wir

werden also gemeinsam eine neue Datei einrichten. Dafür geht man bei DATAMAT einen etwas außergewöhnlichen Weg:

Sie wählen die Funktion Öffnen, als existierte die Datei schon, und geben den gewünschten Namen ein. DATAMAT erkennt, daß diese noch nicht vorhanden ist, und fragt deshalb, ob eine neue erstellt werden soll. Das beantworten wir mit Ja und befinden uns nun im Eingabe-Requester für die neuen Datensätze.

Auf der linken Seite ist eine - noch leere - Liste der Datensätze. Rechts finden wir mehrere Auswahlfelder. Sechs davon sind zur Bestimmung des Feld-Typs. Es werden die Typen Text, Zahl, Datum, Uhrzeit, Auswahl und IFF zugelassen. Bei den IFF-Dateien handelt es sich um Grafiken, die auf Wunsch vom Programm geladen und dargestellt werden können. Man kann jetzt mit der Eingabe beginnen und unter Feldname einen beliebigen Text eingeben, unter dem später das Datenfeld erscheinen soll. So verfährt man für alle Eingabe- oder Datenfelder. Möchte man zwischendurch den Feld-Typ ändern, reicht ein Klick auf eines der dafür vorgesehenen Symbole. Nach Fertigstellung der Grundarbeiten gibt man diese Daten frei. Jetzt will der DATAMAT wissen, ob diese Datei noch mit einem Paßwort belegt werden soll. Dabei gibt es zwei verschiedene Modi: Ein Paßwort zum Nachsehen der Daten, bei dem aber keine Veränderungen gemacht werden können, und ein zweites, mit dem man auch schreibend auf die Daten zugreifen kann. Diese Trennung beider Funktionen macht es möglich, daß auch verschiedene Personen Zugriff darauf haben, die nicht alle berechtigt sind, die Daten zu verändern. Zum Schluß wird diese neu kreierte Datei mit OK quittiert.

Jetzt ist DATAMAT am Zuge und richtet auf der angegebenen Diskette die Datei ein. Das kann je nach Größe einige Sekunden dauern. Er erstellt dazu eine Standard-Eingabe-Maske, bei der es einen Überschrift-Text und eine tabellarische Auflistung aller Datensätze gibt. Damit Sie davon eine Vorstellung haben, sehen Sie in Abb. 3.7 eine Demonstrations-Datei.

Nun können Sie durch einfaches Anklicken eines der Felder unserer neuen Datei mit der Eingabe beginnen. Sie geben dort für einen ersten Test irgendwelche Daten ein und gelangen mit dem Druck auf die [Return]-Taste in das nächste Feld. Mit einem Klick auf das Ausrufungszeichen im Datei-Fenster wird dieser neue Datensatz in die Datei übernommen. Die erste Karteikarte ist beschrieben!

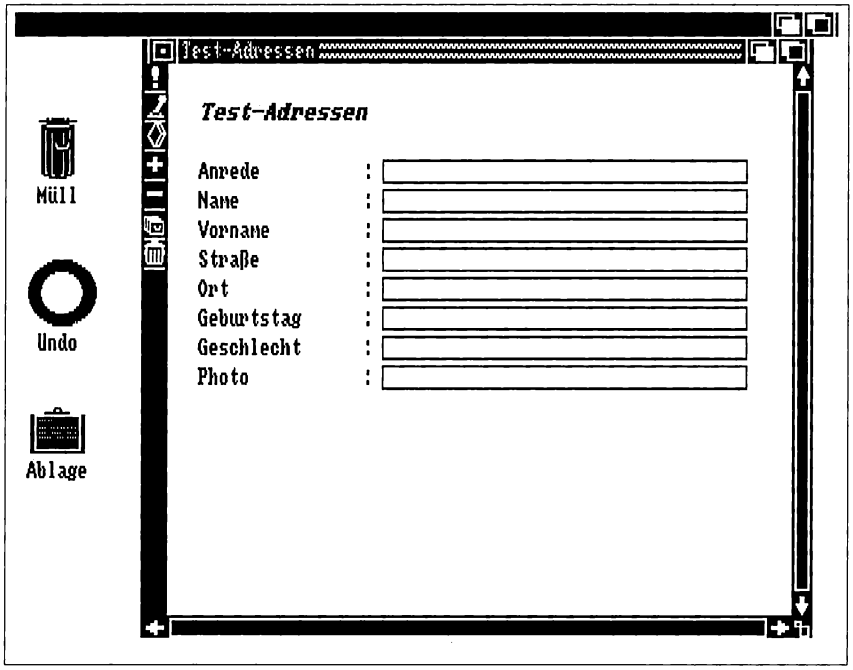


Abb. 3.7: Die Standard-Maske bei jeder Datei

So sollten Sie noch einige weitere Test-Daten eingeben, damit wir weitere Funktionen des Programms testen können. Hat man nämlich mehrere Datensätze angelegt, möchte man oft schnell und einfach einen daraus suchen und ansehen. Dafür gibt es nun zwei Möglichkeiten: Einmal die direkte Suche; dabei wird mit der Maus der Schalter im Window umgestellt, und das Ausrufungszeichen ändert sich in ein Fragezeichen. Jetzt gibt man in den Feldern die Suchkriterien ein, die einem bekannt sind. Die Verwendung von Jokern, mit denen unbekannte Textabschnitte gekennzeichnet werden, ist hierbei zugelassen. Nun kann man DATAMAT suchen lassen, und ein gefundener Datensatz wird dann in der Maske angezeigt.

Die zweite und allgemein übliche Methode ist das Anlegen eines Index. Dafür wählt man diese Funktion im Menü aus und kann festlegen, welches der Datenfelder den Index bereitstellen soll. Von diesem Feld wird nun eine Liste aufgestellt und sortiert. In dieser Liste können Sie mit Hilfe des Karteikasten-

Symbols im Datei-Fenster herumblättern und die gewünschte Datei auswählen, die darauf sofort in der Maske erscheint. Für aufwendiges Suchen lassen sich mehrere Indizes erstellen, mit denen man leicht und schnell in mehreren Listen nachsehen kann.

Betrachten wir jetzt aber wieder die Bildschirmmaske unserer Datei. Sie wurde in dieser Form vom Programm eingerichtet. So ist sie aber bestimmt nicht ansprechend und komfortabel gestaltet. An dieser Stelle zeigt sich der wichtigste Funktionsbereich dieses Programms. Mit dem Masken-Editor kann die Eingabe-Maske, aber auch jede andere Maske, auf die wir später noch kommen, nach jedem Wunsch gestaltet werden. Es ist möglich, die Texte und Felder in ihrer Position zu verändern. Dabei kann jeder Text editiert und neu gestaltet werden. Zur Auswahl stehen alle bekannten Schriftattribute und Zeichensätze, die von der Workbench nachgeladen werden können. Außerdem ist die Farbe frei wählbar. Auch die Datenfelder sind editierbar. Und zwar sind als veränderbare Parameter äußerlich Farbe und Zeichensatz einzustellen. Sehr weitreichend ist die Änderung der Größe, bei der die Höhe, Breite und Gesamtzahl der Zeichen angezeigt werden. Hiermit können Sie z.B. bei Textfeldern das Fassungsvermögen bestimmen und angeben, über wieviel Zeilen der Text gehen soll.

Um die Gestaltung der Bildschirmmaske ansprechender zu machen, ist es neben den Texten für die Datenfelder möglich, auch separate Texte zu erzeugen, die der Erklärung und Information dienen können. Außerdem kann in den Vorder- oder Hintergrund der Maske Grafik gelegt werden, die aus Rechtecken oder Kreisen besteht. Durch freie Farbwahl und Überlagerung dieser Flächen lassen sich interessante und aufwendige Grafiken erzeugen, die einerseits gliedernde Funktion und andererseits auch ergänzend wirken.

Der Masken-Editor ist auch für andere Ausgaben vorgesehen. So gestaltet man damit außer der Bildschirm-Maske, die für die Eingabe und Bildschirm-Ausgabe vorgesehen ist, auch noch eine Formular- und eine Listen-Maske. Über die Formular-Maske kann die Ausgabe für einen Datensatz auf dem Drucker bestimmt werden. Hier müssen nicht alle Felder, die bei der Eingabe verwendet wurden erscheinen. Und auch die Texte - z.B. Erklärungen oder Hinweise - können fehlen oder durch andere ersetzt werden.

Das gleiche gilt für die Tabellen-Maske, bei der die einzelnen Spalten der Tabelle eingestellt werden. Die Größe jedes Feldes kann hier noch einmal ganz separat justiert werden, damit z.B. bei längeren Textfeldern nur die ersten 10

Zeichen erscheinen. Außerdem ist es möglich, bestimmte Spalten aufzusummieren, um somit eine einfache Berechnung durchführen zu können.

Alle drei Maskentypen werden getrennt verwaltet. Es wurde so konzipiert, daß man für jede Anwendung theoretisch eine neue Maske verwenden kann. Es lassen sich sogar mehrere Bildschirmmasken oder die eines anderen Typs definieren, bei denen der grafische Aufbau unterschiedlich gestaltet ist. So kann man auch bestimmte Datenfelder in der einen Maske weglassen, um diese als geheim zu verwalten und nur Eingeweihten zugänglich zu machen.

3.3 Grafikanwendung

Einer der beliebtesten Anwendungszweige des Amiga ist sicherlich die Grafik. Schöne, farbenprächtige und häufig sogar bewegte Bilder lassen das Herz eines jeden Amiga-Besitzers höher schlagen. Doch, wie sieht es überhaupt auf dem Software-Markt aus?

Dieses Kapitel will sich mit der leidigen Frage beschäftigen, welches Grafik-Programm für welche Anwendung das Richtige ist. Da der Amiga ein ausgesprochener Grafik-Computer ist, wird es für den Einsteiger um so schwerer sein, aus der Fülle von Angeboten das Richtige herauszusuchen.

Wir möchten deshalb an dieser Stelle die wichtigsten und besten Programme vorstellen, die auch preiswert und zugleich leistungsstark sind. Der zweite nicht zu vernachlässigende Punkt ist die Art der Grafik. Stellen wir nun zuerst einmal zusammen, welche Fähigkeiten und Leistungen ein Grafik-Programm bieten muß, damit es den Standard an Komfort erfüllt.

- Nicht mehr wegzudenken ist der Standard im Speicherformat. Ohne IFF läuft bei Grafikprogrammen nichts mehr. Einerseits, weil mit der Zeit schon so viele Grafiken in diesem Format vorliegen, daß sich ein Programm einfach als unbrauchbar abstempeln würde, wenn es diese nicht laden könnte, zum anderen ist nur so die Weiterverarbeitung in anderen Programmen sichergestellt.

- Ein zweiter wichtiger Block sind die Zeichen-Funktionen. Sie entscheiden, welche Möglichkeiten das Programm bietet. Hier gibt es neben den Linien, Kreisen, Rechtecken und den gefüllten Varianten noch einigen

Schnickschnack, der bei genauerem Betrachten sehr hilfreich sein kann oder aber vollkommen unbrauchbar ist.

- Als drittes - und das ist nicht zu verachten - möge man auf die Organisation des Programms und seine Bedienung achten. So manches Programm, das über tausend Funktionen anbietet, ist einfach unbrauchbar, weil die Funktionen auf so umständliche Weise angewählt werden müssen, daß ein Zeichnen von Hand schneller wäre.

An dieser Stelle darf man sich aber nicht auf das Althergebrachte verlassen. Nur weil ein Programm die Bedienung über Intuition organisiert, ist es noch lange kein Spitzenreiter. Gerade neue innovative Methoden haben manchmal etwas an sich, das noch mehr Komfort beim Zeichnen ermöglicht.

Wie Sie hieran sehen, ist es gar nicht so einfach, zu entscheiden, wann ein Programm leicht zu bedienen ist. Es gibt keine eindeutigen Kriterien. Deshalb haben wir uns auch an einige Programme gesetzt und diese bei der täglichen Arbeit ausprobiert. Manche Zeichnungen in diesem Buch sind dabei entstanden, und die Erfahrung hat es uns ermöglicht, an dieser Stelle ein Urteil zu fällen.

Wir beginnen den Test der Zeichenprogramme mit dem Klassiker Deluxe Paint III, dessen Name schon in jedermanns Ohr klingt. Die Frage lautet: "Warum muß dieses Programm noch getestet werden?" Und die Antwort darauf heißt: "Um einen Vergleich zu anderen Programmen verständlich zu gestalten und denen, die das Programm noch nicht auf ihrem Amiga laufen hatten, einen Eindruck davon zu vermitteln!"

Deluxe Paint

Das Programm präsentiert sich beim Kauf in einem attraktiven DIN-A4-Karton, den das schon zum Markenzeichen gewordene Bild der Maske von Tut Ench Amun schmückt. Uns lag zum Test die deutsche Version vor, die auch den PAL-Bildschirm ausnutzt.

Im Innern des Kartons findet man das Handbuch, welches als Ringbuch sehr praktisch aufgemacht ist. Leider konnten wir die Seitenzahl nicht genau ermitteln, weil sie in jedem Kapitel neu beginnt. Es sind aber mit Sicherheit

mehr als 100 Seiten, die jede Funktion des Programms sehr ausführlich dokumentieren. Aber zur Anleitung kommen wir später noch.

Weiterhin befinden sich zwei Disketten im Lieferumfang. Auf der einen ist das Programm, die Programm-Disk, und auf der anderen einige Bilder, die Art-Disk. Weder die eine noch die andere ist kopiergeschützt, was es ermöglicht, für die tägliche Arbeit eine Sicherheitskopie anzufertigen.

Allerdings muß man beim Start der Kopie die Orginal-Diskette einlegen, um sich als rechtmäßiger Benutzer zu identifizieren. Auch eine Registrierkarte ist mitgeliefert, die aber keine Seriennummer benötigt, sondern eher statistische Angaben zum Kauf und zur Person fordert.

Das Programm starten

Hat man sich über seine Workbench eine Sicherheitskopie angefertigt, kann man über die Programm-Diskette von Deluxe Paint das System booten. Hardware-Anforderungen für das Programm sind mindestens 512 KByte Speicher und ein Diskettenlaufwerk. Je mehr Speicher, desto besser, denn nur dann können auch größere Grafiken gezeichnet werden, weil Deluxe Paint seine Programmteile dann in dem Speicherbereich ablegt, in dem keine Grafiken gespeichert werden können.

Ist das Programm über das Icon gestartet, erhält man die erste Auswahl-Tafel, in der man nach der Auflösung und der Anzahl der Farben gefragt wird. Weiterhin kann man bestimmen, ob immer nur Teile des Programms geladen werden sollen.

Das ermöglicht es, mehr Grafik zu bearbeiten, dafür aber bei der Auswahl eines neuen Befehls eine Wartezeit in Kauf zu nehmen, die durch das Nachladen des Befehlssegmentes entsteht.

Vor uns finden wir jetzt die Grund-Fläche von Deluxe Paint. Auf der rechten Seite ist eine senkrechte Leiste, auf der die wichtigsten Funktionen des Programms und die Farbpalette untergebracht sind, in der Titelzeile des Screens finden wir Angaben zum Programm und zu den Einstellungen. Diese sind aber noch nicht aktiv. Der Rest des Bildschirms bleibt frei und kann als Zeichenfläche genutzt werden. Sie können sich das jetzt einmal ansehen:

Abb. 3.8: Deluxe Paint und die Aufteilung des Bildschirms

Bevor wir nun zu zeichnen beginnen - das Programm wartet ja förmlich darauf - sehen wir uns erst einmal die beigefügte Anleitung an:

Das Handbuch

Allein schon der äußere Eindruck des Handbuchs ist sehr gut. Das Buch ist mit Drahtringen gebunden, so daß es beim Aufschlagen einer Seite - man möchte parallel dazu etwas am Amiga ausprobieren - nicht wieder zusammenklappt, wie es bei geleimten Büchern oft der Fall ist. Kommen wir aber jetzt zum Inhalt des Buches selbst.

Die Dokumentation teilt sich in vier große Kapitel auf, zu denen sich noch ein Anhang gesellt. Nach einer Einführung über die Computergrafik allgemein und dieses Handbuch beginnt das erste Kapitel mit einer Übersicht der wichtigsten Bedienungsteile. Dies ist deshalb besonders praktisch, weil auf 20 Seiten das zusammengefaßt wurde, was gerade zum Arbeiten ausreicht. So kann der Einsteiger gleich die ersten Erfolge vermelden, wenn er die Beispiel-Grafiken lädt oder mit dem Pinsel auf dem Bildschirm malt. Allerdings reicht

diese Übersicht natürlich nicht aus, um alle Funktionen des Zeichenprogramms effektiv auszunutzen.

Deshalb geht das zweite Kapitel Stück für Stück auf jedes Bedienelement ein. Mit vielen Grafiken werden die Request-Boxen erklärt und alle Einstellungen anhand von einigen Beispielen aufgeführt. Man findet eine Beschreibung der verschiedenen Amiga-Auflösungen und auch die verschiedenen Parameter der Farbpalette dokumentiert.

Genauso groß wie der Nachschlage-Teil ist auch das dritte Kapitel, nämlich 40 Seiten. Hier geht es um Praxis, d.h. man hat sich bei Electronic Arts gesagt, daß es besser ist, die Bedienung eines Programms durch Arbeiten zu lernen, als durch das bloße Aufzählen von Funktionen. In diesem Abschnitt werden in vier Lektionen die Funktionen von Deluxe Paint III gelehrt, die das Programm besonders auszeichnen. So geht man auf die Farbpalette und die komplexen Einstellungen, die Brushes und damit verbundenen Muster, die Masken und das Fixieren des Hintergrunds sowie die perspektivische Darstellung der Grafik ein.

Der Nachschlageteil ist mehr für den fortgeschrittenen Bediener des Programms gedacht. Hier findet man in kurzen, fast tabellarischen Erklärungen noch einmal zu jedem Menüpunkt eine Beschreibung. Teilweise wiederholen sich hier die Informationen, aber das läßt sich nicht vermeiden. Interessant ist die Zusammenstellung der Tastatur-Befehle, über die man sehr schnell Programm-Funktionen aufrufen kann. Leider kann man diese nicht aus dem Handbuch entnehmen, um sie neben die Tastatur zu legen.

Arbeiten mit Deluxe Paint

Kehren wir wieder zum Programm selbst zurück und sehen uns die Grundfunktionen an. Neben den Zeichenmodi für Stecken, Freihand, Kreise und Vierecke werden alle nur denkbaren Körperformen über eine besondere Funktion unterstützt, die so lange Linien miteinander verbindet, bis man mit der Maus wieder am Ausgangspunkt angelangt ist. Diese Flächen können natürlich ausgefüllt werden; entweder mit der Füll-Funktion oder einem eingebauten Modus, der dies gleich beim Zeichnen erledigt. Besonders hervorzuheben ist die Möglichkeit, daß für jeden Zeichenvorgang jede beliebige Grafik als Pinsel genommen werden kann. Wir sind also nicht auf verschieden große

Punkt-Pinsel beschränkt, sondern es können ganze Grafiken beliebiger Form aus Bildern entnommen und zum Weiterzeichnen verwendet werden. Daß dies nicht einfarbig sein muß, braucht wohl nur am Rande erwähnt zu werden.

Die von Deluxe Paint angebotene Farbpalette kann in ihrer Größe nach Auflösung und Speicherplatz frei bestimmt werden. Je nach Anzahl erscheinen alle Farben am rechten unteren Rand. Die Auswahl erfolgt ganz einfach mit der linken Maustaste für die Zeichenfarbe und mit der rechten für die Hintergrundfarbe. Möchte man die Farbwerte verändern, genügt die Auswahl des Punktes Farbpalette im Hauptmenü oder ein Druck auf die Taste [P]. Es erscheint eine Request-Box, in der alle Farben und viele Regler zum Einstellen untergebracht sind. Jede Farbe kann in ihren durch die Technik bedingten Werten von Rot, Grün und Blau bestimmt werden. Außerdem gibt es noch eine Einteilung in Helligkeit und Verhältnis zwischen Farbbereichen. Nicht zu unterschätzen ist die Möglichkeit, Bereiche der Palette zyklisch in beliebiger Geschwindigkeit rotieren zu lassen und so Bewegung ins Bild zu bringen. Manche Programme täuschen so eine aufwendige grafische Veränderung vor, die in Wirklichkeit nur durch Farbvertauschung entsteht.

Die oben angesprochenen Brushes - so nennt man die Pinsel, mit denen gezeichnet werden kann - sind aber nicht nur frei aus der Grafik zu wählen. Es gibt ein extra Menü für sie, in dem man die Pinsel laden oder speichern kann. Weiterhin sind viele Modifikationen unterstützt wie das Drehen, Spiegeln oder Verzerren. Im Ganzen läßt sich eigentlich alles mit den Brushes machen, was man sich nur vorstellen kann. Das kann man auch erwarten, denn sie spielen auch als Füllmuster, Hintergrund oder eben als Zeichenstift eine besondere Rolle.

Insgesamt hat Deluxe Paint III sechs verschiedene Menüs, die in der uns vorliegenden Version alle eingedeutscht wurden. Das erste Menü beherbergt die Grundfunktionen. Laden, Speichern und das Ausdrucken der Grafik werden hier mit Funktionen angeboten. Für die Disketten-Arbeit besitzt Deluxe Paint eine File-Request-Box mit allem nötigen Komfort. Der Ausdruck kann neben den Einstellungen über Preferences noch mit einigen Werten im Programm selbst beeinflußt werden. Auch dafür erscheint eine Request-Box. Andere Menüpunkte beziehen sich auf die Grundeinstellungen, so das Format der Grafikseite, die Farbeinstellungen und die zweite Seite, auf der Hilfsgrafiken zwischengespeichert oder getestet werden können.

Das zweite Menü, das für die Pinsel, hatten wir oben schon besprochen. Im dritten Menü wird der Modus festgelegt, in dem Deluxe Paint zeichnen wird. Neben dem ganz normalen Linen-Ziehen mit dem eingestellten Pinsel kann man Schmieren, Schattieren, Mischen, Rollen und Weichzeichnen. Mit dem Weichzeichner versucht Deluxe Paint unter Verwendung der vorhandenen Farben weichere Übergänge zu erzeugen. Das Rollen läßt einfach die Zeichenfarbe nach der Farbpalette wechseln, so daß interessante Muster entstehen, die besonders schöne Wirkungen haben, wenn man dann die Farben zyklisch durchlaufen läßt. Sehr witzige Effekte erzielt die Schmier-Funktion, die einfach den Bereich des Pinsels hin- und hertauscht; damit werden Konturen unordentlich und das gesamte Bild verliert seine scharfe Computer-Charakteristik. Das ist zur Unterstreichung des Natürlichen einer Zeichnung bestimmt erwünscht, artet aber zum Entstellen eines Kunstwerks aus, wenn man zu große Bereiche damit bearbeitet. Aber wir verfallen hier in künstlerische Aspekte und wollen eigentlich das Programm testen.

Kommen wir nun zum fünften Menü mit dem Namen Effekte. Es ist dreigeteilt in die Menüpunkte Maske, Hintergrund und Perspektive. Alle drei besitzen weitere Untermenüpunkte, auf die wir jetzt eingehen wollen. Und zwar testen wir als erstes die Masken-Funktion, die sehr detailliert im Handbuch beschrieben ist: der Sinn, die Vorteile und Organisation werden ausführlich besprochen. Allerdings zeigt sich hier ein Nachteil der deutschen Übersetzung. Alle Funktionen im Programm wurden mit deutschen Worten belegt, und auch das Handbuch ist ins Deutsche übertragen worden. Der Übersetzer des Handbuchs hat aber, und das würde ihm jeder gleichtun, die Fachbegriffe im Englischen belassen, was sich leider nicht mit dem Programm deckt. Wenn also im Buch steht, man soll die Option Stencil und im sekundären Menü Make auswählen, muß man dies (vielleicht durch Raten) übersetzen und darauf kommen, daß eigentlich das Menü Maske mit dem Punkt Def. gemeint ist. Gar nicht so einfach!

Sehr wirkungsvoll ist hingegen die Funktion selbst. Mit ihr lassen sich Bereiche ausgewählter Farben einfrieren, und diese sind somit vor weiterem Übermalen gefeit. Erst nach dem Aufheben der Funktion kann wieder alles verändert werden. Der Zweck dieser Übung ist ein großer Vorteil beim Malen. Wir können nämlich jetzt (besonders bei perspektivischen Bildern) zuerst die Dinge zeichnen, die im Vordergrund liegen. Dann werden die Farben eingefroren und man geht weiter in die Tiefe usw. Für die Arbeit mit den Masken werden noch weitere Funktionen wie Laden, Speichern, Ein- und Ausschalten, Invertieren und Sperren angeboten. Deluxe Paint hat hier wirklich nichts ver-

gessen. Genau umgekehrt verfährt die Hintergrund-Funktion. Sie hält die Zeichnung so, wie sie ist, fest und läßt es aber zu, daß man sie übermalt.

Die Auswirkungen werden erst bemerkt, wenn man jetzt die Lösch-Funktion aus der seitlichen Leiste auswählt. Der Hintergrund, also das, was wir als Hintergrund vorhin definierten, bleibt erhalten, und nur das darüber Gezeichnete verschwindet. Man kann so Veränderungen austesten und bei Nichtgefallen widerrufen.

Ein großes Gebiet der Fähigkeiten von Deluxe Paint sind sicherlich die Perspektiv-Funktionen, die wir im dritten Menüpunkt finden. Allerdings haben wir hier ein so großes Spektrum vor uns, daß wir nur die Fähigkeiten beschreiben wollen, nicht aber die Arbeit. Denn Sie können sich vorstellen, daß selbst bei einem komfortablen Programm einige Arbeitsschritte nötig sind, bis man eine perspektivische Zeichnung erreicht hat.

Zum Abschluß sehen wir uns noch das Options-Menü an, das als letztes in der Menüleiste auf der rechten Seite zu finden ist. Es enthält einige bemerkenswerte Funktionen, die hier nicht unerwähnt bleiben sollen.

Zuerst einmal lassen sich die Koordinaten beim Zeichnen einblenden. Sie werden dabei in der Menüleiste untergebracht. Die Ausgabe ist sehr schnell! Zur Beschleunigung des Zeichnens gibt es auch eine spezielle Funktion. Beim Ziehen von Linien wird der Pinsel ja immer wieder neu benutzt. Das macht sich zeitlich besonders bemerkbar, wenn es sich um einen grafisch sehr aufwendigen Brush handelt und nicht nur um ein Pixel. Mit Anwahl von "schnell" zeichnet Deluxe Paint hier so lange nur eine kontrastreiche Linie, bis man die Linie oder die Figur endgültig haben will. Erst dann wird das richtige Zeichen-Objekt verwendet.

Weil Deluxe Paint den Workbench-Screen ausschaltet, wenn dies möglich ist, um möglichst viel Grafik-Speicher, sprich Chip-RAM, frei zu haben, gibt es hier eine Funktion, die die Workbench wieder einschaltet bzw. auf Wunsch ausschaltet. Dies ist sicherlich besser, als wenn man während der Arbeit mit dem Programm nicht auf der Workbench arbeiten könnte. Schließlich ist der Amiga ein Multitasking-Computer.

Damit haben wir einen Überblick über die vielfältigen Funktionen dieses Zeichenprogramms gewonnen. Sicherlich ist dieser Eindruck nicht vollständig, aber wir können ihn durchaus positiv nennen. Dem Programm fehlt es an kei-

ner Möglichkeit, die man vermissen könnte. Es ist eher so, daß durch die Zeit, die seit der Programmierung vergangen ist, einige Funktionen etwas mehr an Komfort erwarten lassen, aber dies trübt das Bild fast gar nicht.

Das ausführliche Handbuch und die große Anzahl an Möglichkeiten machen das Programm zum Standard der Zeichenprogramme. An dieser Stelle soll aber auch nicht verschwiegen werden, daß man für weniger Geld ähnliche Programme bekommt, die zwar nicht alle Leistungsmerkmale besitzen, sich aber am internationalen Standard orientieren. Hier muß man sich die Frage stellen, ob es nötig ist, Funktionen zur perspektivischen Gestaltung zu haben oder ob es für den alltäglichen Bedarf nicht ausreicht, wenn man die Grundfunktionen ausnutzt.

Diese Fragen haben wir uns auch gestellt und deshalb eines der neuesten Zeichenprogramme getestet. Es liegt preislich bei 100,- DM und ist etwas weniger üppig ausgestattet. Sehen Sie selbst:

The Graphics Studio

The Graphics Studio ist in einem etwa DIN A5 großen und Daumenbreite-hohen Karton verpackt. Darin finden wir neben etwas Schaumstoff zum Ausfüllen die 50seitige Anleitung und die Programmdiskette. Das Programm erfordert ein Diskettenlaufwerk und kommt mit 512 KByte Speicher aus. Es läuft auf allen Amiga, also insbesondere auf dem 500er. Die Diskette ist nicht kopiergeschützt, und es wird gleich zu Anfang empfohlen, das Programm nur über eine Sicherheitskopie zu betreiben, damit man nicht aus Versehen die Orginal-Diskette zerstört. Das alles wird für 99,- DM geliefert.

Das Handbuch

Die Anleitung ist mit drei Kapiteln und einem Anhang ziemlich kurz gehalten. Da sie aber in deutscher Sprache verfaßt wurde und keine Funktion des Programms ausläßt, kann man sie als ausreichend bezeichnen.

In der Einleitung wird darauf hingewiesen, daß das Zeichenprogramm so konzipiert sein soll, daß man es ohne große Mühe erlernen kann. Dafür wurde die gesamte Bedienung darauf ausgerichtet, mit Symbolen zu arbeiten. Es werden

nur ganz wenige Funktionen in Text-Menüs angeboten. Vielleicht ist es auch darin begründet, daß die Anleitung so kurz gehalten wurde.

Nachdem das Starten des Programms beschrieben ist, findet man eine Abbildung vor sich, mit deren Hilfe zuerst der allgemeine Aufbau des Programms beschrieben wird. Dann geht die Anleitung auf die elementaren Funktionen ein. Die Farbpalette wird an dieser Stelle leider nur in ihrer Auswahl beschrieben. Erst viel später, im dritten Kapitel, geht das Handbuch darauf ein, wie man sie verändert und die Sonderfunktionen auswählt. So sind die meisten Beschreibungen nach der Schwierigkeit gesteigert.

Zu allen weiteren Funktionen findet man immer das Symbol, mit dem man sie auswählen kann, und dazu eine knappe Beschreibung. An dieser Stelle findet man leider keine Übungen, die das Arbeiten verdeutlichen könnten. Hier zeigt sich, warum das Programm preiswerter angeboten wird. Allerdings stellt sich auch die Frage, ob man ausführlich erklären muß, wie man eine Linie mit der Maus zeichnet. Es wird jedem klar sein, daß dazu auf der Zeichenfläche die linke Maustaste gedrückt werden muß, und schon setzt das Programm die Punkte. Eigentlich reicht die Beschreibung in der Anleitung aus, wie man die Funktion anwählt.

Das zweite Kapitel geht auf komplexere Teile des Programms ein. Es beschreibt die einzelnen Zeichenmodi und erklärt deren Anwendung und Auswirkungen. Außerdem werden hier die speziellen Eigenschaften des Malprogramms erwähnt und die Block-Operationen aufgezählt. Es gibt eine Random-Color-Funktion, mit der die Zeichenfarbe zufällig ausgewählt wird. In gleicher Weise unterstützt das Programm auch die Schattendarstellung, bei der zu jedem Objekt einfach ein Schatten in einer beliebigen Verschiebung gezeichnet wird. Die Block-Operationen des Programms sind vielfältig. Ein rechteckiger Bereich beliebiger Größe kann aus dem Bild entfernt und an anderer Stelle wieder in das Bild eingesetzt, gedreht, gespiegelt, gestaucht oder gestreckt sowie gelöscht werden. Das ist im Vergleich als Ersatz zu den Brush-Funktionen von Deluxe Paint zu sehen.

Die ersten Schritte mit dem Programm

Man startet das Programm, indem es anstelle der Workbench-Diskette eingelegt wird. Von der Workbench ließ es sich leider nicht immer zum Arbeiten

bewegen. Nach einigem Laden und nach dem Titelbild erscheint die Arbeitsfläche. Um jetzt ein Bild zu laden, geht man mit dem Maus-Cursor auf die Titelleiste und drückt dort die rechte Maustaste nieder. Im File-Menü wählt man die Funktion zum Laden aus, und wir bekommen eine riesige File-Request-Box auf den Bildschirm. In ihr wird das Verzeichnis der Diskette dargestellt. Durch einfaches Anklicken des File-Namens wird das entsprechende Bild geladen. Dabei akzeptiert das Programm schon hier beide Auflösungsstufen, die unterstützt werden. Sowohl Low-Res (320 mal 200) als auch Hi-Res (640 mal 200) können dargestellt werden. Dabei transformiert The Grahpics Studio das Bild in das jeweilige Format, indem es Punkte doppelt setzt oder wegläßt. Man erkennt schon an dieser Stelle, daß The Graphics Studio ein amerikanisches Programm ist. Denn wir haben nicht den vollen PAL-Bildschirm zur Verfügung. Nur 200 Zeilen werden für die Darstellung der Grafik genutzt. Dafür ist es durchaus möglich, Bilder zu zeichnen, die doppelt oder dreifach so groß sind. Der Bildschirm zeigt dann immer nur einen Ausschnitt, der über die Maus verschoben werden kann.

Die Funktionsgruppen

Der Bildschirm unterteilt sich, wie Sie oben sehen können, in drei Bereiche. In der Mitte finden wir in einem Fenster einen Ausschnitt der Grafik (bei normalen Grafiken verdecken nur die obere und untere Leiste einen Bildbereich). Oben ist dann die Menüleiste mit der Status-Anzeige. Gleich darunter ist ein dreigeteilter Bereich, der auf der linken Seite Funktionen zur Texteingabe, zum Flächen-Füllen, für die Block-Operationen und zum Löschen enthält. In der Mitte dieses Bereiches sind eine ein- und ausschaltbare Koordinatenangabe (das Zeichnen verlangsamt sich etwas, wenn man die Koordinaten ausgeben läßt), das Undo-Gadget und ein Schalter für die zweite Zeichenseite.

Rechts daneben sind die Zeichen-Tools. Die erste Leiste gestattet die Auswahl des Zeichenmodus: durchgezogene Linien, Punkt-Linien oder Zoom, mit dem Details des Bildes bearbeitet werden können. In der zweiten werden Linien angeboten. Die Auswahl geht von einfachen Strecken hin zu verbundenen und radial verlaufenden. Mit den beiden anderen Klick-Leisten werden Rechtecke und Kreise angeboten. Auch hier sind viele verschiedene Möglichkeiten angeboten. Sie reichen von Objekten, bei denen nur der Rand gezeichnet wird, über Objekte mit gleichem Mittelpunkt hin zu den ausgefüllten Kästen bzw. Kreisen.

Im unteren Bereich haben wir es hauptsächlich mit der Farb- und Musterauswahl zu tun. Ganz rechts ist eine Auswahl der vordefinierten Muster bereitgestellt. Diese können jederzeit editiert, d.h. verändert, oder aus der Grafik herausgegriffen werden. Dazu finden wir dann die Farbpalette, bei der Sie über einen einfachen Klick die Zeichenfarbe einstellen können. Was gezeichnet wird, läßt sich links neben der Farbpalette ablesen. Entweder steht hier die Standard-Vorder- und Hintergrundfarbe, oder es wird das Muster angezeigt, mit dem man gerade zeichnet. Auch die Hintergrundfarbe oder das Zeichenmuster wird über einfaches Anklicken bestimmt. Selbst beim Punktzeichnen wird das Muster wiedergegeben. Der Bildschirm ist dann quasi unsichtbar mit dem Muster überdeckt, und beim Zeichnen wird das Muster an der entsprechenden Stelle sichtbar. Man muß nur lange genug zeichnen, um das gesamte Muster zu Gesicht zu bekommen. Auf der linken Seite des unteren Bereiches sind drei Auswahlmenüs für den Zeichenpinsel. Es gibt eckige, runde und linienartige in mehreren verschiedenen Stärken.

Am linken Rand sieht man schließlich eine Scroll-Einrichtung. Neben zwei Pfeil-Gadgets, mit denen der Ausschnitt, der in der Mitte zu sehen ist, verschoben werden kann, ist gleich daneben eine Darstellung des Ausschnitts im Gesamtbild. Wie man sieht, kann ein Bild leider nur in vertikaler Richtung größer gezeichnet werden. Hierbei ist aber mehr als das Vierfache gestattet, wenn es der Speicher zuläßt.

Die Zeichen-Tools

Interessant sind die Möglichkeiten, die hier von The Graphics Studio angeboten werden. Neben dem Standard wie der zusammenhängenden Linie und der unterbrochenen, den Strecken, Vierecken und Kreisen gibt es auch noch unbekanntere Varianten. So lassen sich die Vier- oder Rechtecke mit abgerundeten Ecken zeichnen. Eine Arbeit, die sonst viel Tüftelei erforderte, wird so zum Kinderspiel. Das Programm unterstützt auch das Zeichnen von mehreren Kreisen oder Rechtecken mit dem gleichen Mittelpunkt. Das erleichtert es, einen Ring oder Rahmen zu konstruieren. Etwas dürftig ist der Zoom-Modus ausgefallen. Er erlaubt drei festgelegte Vergrößerungen, aber nur das Zeichnen in zwei Modi: Punkte oder Flächen füllen. Andere Tools werden nicht unterstützt.

Die Special-Tools

Auf der linken Seite in der oberen Zeile finden wir die speziellen Tools. Hier bieten sich Funktionen für Texte, zum Füllen in verschiedenen Modi und Block-Operationen, mit denen sich größere Teile der Grafik mit komplexen Funktionen beeinflussen lassen.

Wählt man ein T für Texte an, kann mit der Maus der Cursor in der Grafik positioniert werden, und man kann mit der Texteingabe beginnen. Der Text kann sowohl blockweise als auch transparent mit der Grafik verknüpft werden.

Außerdem stehen Manipulationen wie Unterstreichen, Fett- oder Kursivdruck zur Verfügung. Der fertige Text läßt sich schließlich mit wählbaren Bezugspunkten zentrieren und auch als Pinsel verwenden, wenn man besondere Effekte erzielen will. Natürlich stehen alle Zeichensätze der Workbench zur Verfügung.

Interessant sind die verschiedenen Füll-Möglichkeiten, die vom Graphics Studio angeboten werden. Neben der bekannten Art, eine Fläche auszufüllen, bei der die Zeichenfarbe verwendet wird und die gesamte Fläche in dieser erscheint, gibt es auch zwei Varianten, die mit farbigen Linien füllen.

Es ist möglich, zwischen vertikalen und horizontalen Linien zu wählen, so daß sich Raster zur Kenntlichmachung leicht in Schwarzweiß-Grafiken erstellen lassen. Ebenfalls unterstützt werden Polygone, das sind Flächen mit einer bestimmten Anzahl von Eckpunkten, die in festgelegter Reihenfolge mit Linien verbunden werden. Diese Fläche füllt die Funktion bei Erreichen des Anfangspunktes auch in der gewünschten Farbe oder dem Muster aus.

Besondere Aufmerksamkeit sollte man auf die Copy-Tools richten, mit denen Ausschnitte der Grafik beeinflußt werden können. Neben dem einfachen Verschieben und Löschen eines Bereiches gibt es noch weitere Funktionen, mit denen sich diese Ausschnitte beeinflussen lassen.

Es gibt die Möglichkeit, einen Ausschnitt an beiden Achsen zu spiegeln oder um eine beliebige Gradzahl zu drehen. Außerdem kann der markierte Bereich vergrößert oder verkleinert werden, wodurch auch Verzerrungen möglich sind, wenn man nicht gleichmäßig in X- und Y-Richtung dehnt.

Ein Fazit

Sollte man sich an dieser Stelle entscheiden, welches der Zeichenprogramme das bessere sei, so ist eine eindeutige Entscheidung nicht unbedingt möglich. Durch die Vielfalt der von The Graphics Studio angebotenen Funktionen wertet sich das Programm auf, und der geringe Preis tut ein weiteres dazu. Es ist für den Anwender wichtig, zu wissen, welche Erwartungen er an ein Zeichenprogramm stellt. Will man professionell Computergrafik erstellen, so kommt man um Deluxe Paint nicht herum, denn es bietet gerade für die komplizierte Arbeit einige sehr hilfreiche Funktionen. Der einfache Anwender aber, der mit dem Zeichenprogramm vielleicht nur fertige Grafiken bearbeitet oder nur sehr selten ganze Bilder zeichnet, sollte mit dem Graphics Studio gut auskommen. Zumal es um einiges preiswerter ist und doch viele Funktionen in petto hat.

Calligrapher

Der Ausbau der Grafikfähigkeiten des Amiga nimmt immer mehr zu. So waren höhere Auflösungen früher noch ein großes Problem, doch heute gibt es Erweiterungskarten, die dieses spielend meistern. Ebenso ist es mit den Schriften beim Amiga. Zur ersten Version gab es nur wenige Zeichensätze mitgeliefert. Erst ab 1.2 wurde die Gestaltung vielfältiger. Und nun, wo die Version 1.3 schon in vielen Amiga installiert ist, werden die Ansprüche immer höher gesteckt.

Was liegt da näher, als sich um Software zu bemühen, mit der man einfach und ohne große Umstände eigene extravagante Zeichensätze erstellen und entwerfen kann. Zwar wird auf der Extras-Diskette ein Zeichensatz-Editor mitgeliefert - er heißt "Fed" (für Font Editor), doch kann dieser nicht ganz überzeugen. Das liegt hauptsächlich an den vom Programm gesetzten Grenzen, die es nur erlauben, mit einer Farbe zu arbeiten, und die Größe der Zeichen auf Punkte einschränken.

So muß man wohl oder übel auf einen anderen Editor ausweichen, mit dem alle Forderungen befriedigt werden. Dann kommt man zwangsläufig zu "Calligrapher", einem, wie es sich selbst nennt, "Multicolor font editor", der wirklich keine Fähigkeit des Betriebssystems, die uns bis jetzt bekannt ist, unberührt läßt.

Lieferumfang

Man bekommt dieses Programm für ca. 198,- DM. Es wird mit einem mittelstarken, solide verpackten Handbuch und einer Diskette geliefert. Auf dieser Diskette befinden sich der Editor, das Programm "Color Text", mit dem es auch unter 1.2/1.3 möglich ist, farbige Fonts zu verwenden, und einige Demonstrations-Zeichensätze bzw. Grafiken.

Als positiver Punkt ist gleich am Anfang zu nennen, daß dieses Programm keinerlei Kopierschutz enthält und es so ohne Probleme von der Workbench aus möglich ist, eine Arbeitskopie anzufertigen. Natürlich ist es so genauso einfach, das Programm auf einer eventuell vorhandenen Festplatte zu installieren.

Nach dem Programmstart findet man folgenden Bildschirm vor sich:

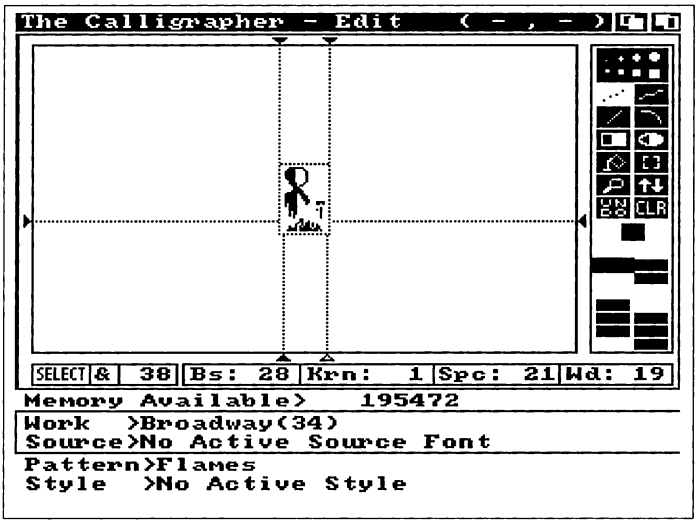


Abb. 3.9: Calligrapher

Nun kann der erste Schritt zu einem neuen Zeichensatz unternommen werden.

Dafür eignet es sich am besten, einen der mitgelieferten Fonts zu verändern und daran die Funktionen des Programms kennenzulernen. Man wählt also im

ersten der drei Menüs die Funktion "Font laden" und geht damit in die Zeichensatz-Verzeichnisse auf der mitgelieferten Diskette.

Nach Auswahl eines Namens in der komfortabel gestalteten File-Request-Box kommt man in das entsprechende Verzeichnis und kann dort die Punktgröße anklicken. Für diejenigen, denen die Zeichensatzorganisation nicht bekannt ist, sei gesagt, daß verschiedene Zeichensatznamen für unterschiedliche Schrifttypen stehen. Unter diesem Namen findet man ein Verzeichnis, in dem die Grafik-Daten unter Nummern abglegt sind. Diese Nummern entsprechen der Größe der Zeichen in Punkten.

Danach ist es möglich, diesen angewählten Zeichensatz, der nun im unteren Bildschirmteil in einer Liste aufgeführt ist, zu bearbeiten. Diese Liste verwendet das Programm, um mehrere Fonts gleichzeitig zu bearbeiten!

Man aktiviert seinen Namen durch Mausklick und sieht alle Einstellungen in einer darüber liegenden Box. Im oberen Drittel ist es nun gestattet, einen Text einzugeben, an dem man die Schriftart untersuchen kann. Durch neues Anklicken eines anderen Zeichensatzes in der Tabelle wechselt das Aussehen des Probetextes blitzschnell.

Hat man den gewünschten Zeichensatz ausgewählt, so kann die Bearbeitung beginnen. Dazu bietet das zweite Menü insgesamt 6 Bereiche an, in denen Einstellungen oder Veränderungen am Zeichensatz durchgeführt werden können.

Neben den Einstellungen auf dem Hauptscreen, die sich auf die Abmessungen und die Farbe beziehen, läßt sich in den Funktionen Edit, Effects, Style und Pattern das Aussehen des Zeichensatzes bestimmen.

So dient der Unterpunkt Edit für die grundsätzliche grafische Gestaltung des Zeichensatzes. Hier kann man mit einem Editor, der dem Zeichenprogramm Deluxe Paint - wenn auch in abgespeckter Version - sehr ähnlich ist, jeden Buchstaben Pixel für Pixel editieren. Dafür findet man an der rechten Seite die Grund- und Zeichenfunktionen wie das Zeichen von einfachen oder zusammenhängenden Punkten und das Zeichen von Linien oder Bögen. Für aufwendigere Arbeiten bietet das Programm noch Kästen und Kreise an, die man ausgefüllt oder nur als Rahmen aufrufen kann. Eine separate Füllfunktion steht ebenfalls zur Verfügung. Selbst das Vergrößern des Arbeitsbereiches mit

einer Lupe ist möglich. Zum Kopieren von wiederkehrenden Zeichenelementen ist es möglich, rechteckige Bereiche auszuschneiden und zu kopieren.

Mit Effects läßt sich nicht nur ein normal bestehender Zeichensatz hinsichtlich der Darstellung in Fettschrift, Kursivschrift oder anderen Merkmalen verändern. Hier werden auch die Parameter beeinflußt, die den Lückenabstand, die vertikale Position des Zeichens oder aber die BitMap für die Grafik festlegen. Damit kann der Charakter noch einmal wesentlich verändert werden.

Style hat nichts mit den schnöden Effekten zu tun. Was man hier mit dem Zeichensatz anstellen kann, ist beeindruckend. Hier läßt sich ein Zeichensatz aus bis zu 16 Layern (hintereinander liegenden Grafiken) zusammenstellen, die einzeln in Farbe, Position und Aussehen beeinflußt werden können. Damit lassen sich Zeichensätze erstellen, wie man sie bisher noch nicht gekannt hat.

Die Welt der "Patterns" (Muster) ist bestimmt schon aus Zeichenprogrammen bekannt. Auch hier haben bekannte Zeichenprogramme eine nicht unbedeutende Rolle gespielt. Bei den Fonts werden die Patterns dafür benutzt, eine Grundlage zum Zeichnen und auch für den Hintergrund zu geben. Im Ganzen läßt sich sagen, daß dieses Programm alle Funktionen bietet, die man zum Überarbeiten oder Neugestalten eines Zeichensatzes benötigt. Zwar erkennt man an einigen Stellen, daß die Bedieneroberfläche nicht alle Features von Intuition nutzt und das Design nicht immer ansprechend ist. Dies wird aber von den Fähigkeiten weitaus übertroffen.

Mitgeliefert wird übrigens ein sehr nützliches Tool, das es von der Workbench aus ermöglicht, das Verzeichnis der Zeichensätze von der Boot-Diskette auf ein beliebiges Laufwerk zu legen.

Software-Überblick:

Programm:	Calligrapher Zeichensatzeditor
Preis:	DM 198,-
Version:	V II 1.00
Lieferumfang:	Handbuch mit 80 Seiten, 2 Disketten
Hardware:	mind. 512 KByte
Kickstart:	ab 1.2

Dieses Programm bietet also, obwohl es in seiner Benutzeroberfläche sachlich nüchtern gestaltet ist, überaus viele Möglichkeiten zur anwenderunterstützten Zeichensatzveränderung oder Entwicklung. Man möchte sagen, daß es für den Grafik-Freak genauso zum Standard gehört wie das Zeichenprogramm Deluxe Paint.

3.4 Sound- und Klang-Anwendungen

Genauso faszinierend, wenn auch (leider) viel weniger beachtet, sind die Klangspiele, die man mit dem Amiga veranstalten kann. Nichtsdestotrotz gibt es natürlich auch auf diesem Gebiet Software. Schließlich sind vier Soundkanäle, die in Stereo-Manier angesteuert werden können, es wert, mit Software bedacht zu werden. Digitaler Klang ist kein Zauberwort mehr und Reinheit kein unerreichtes Ziel, seit wir nicht nur mit Rauschkanälen auskommen müssen.

Um nun den Anwendungen im Gebiet der Klänge gerecht zu werden, haben mehrere Software-Firmen Programme herausgebracht, mit denen es auf verschiedenste Weise möglich ist, dem Amiga Töne zu entlocken. Allerdings stehen wir schon hier vor einem Problem. Das Lager teilt sich nämlich an dieser Stelle in zwei Gruppen.

Die einen legen Wert auf Musik im herkömmlichen Sinne. Das heißt, es wird versucht, über die Eingabe von Noten Musikstücke vom Amiga nachspielen zu lassen. Dies mit einer beliebig wählbaren Instrumentation und einigem Komfort, aber es ist eben Noten-Musik.

Andere wiederum gehen ganz andere Wege. Sie lassen vom Amiga synthetische Klänge erzeugen und reihen diese in verschiedenster Weise aneinander. Diese Klänge werden über die Eingabe von Wellenformen erreicht oder aber durch Digitalisieren in den Amiga eingelesen und dort weiterverarbeitet. Sie sehen schon, wir haben hier einen sehr deutlichen Unterschied.

Mit der dritten und letzten Variante liegen wir genau in der Mitte beider Interpretationen der Computer-Musik. Diese User streben nämlich die möglichst naturgetreue Nachbildung von bekannten Klängen, das sind meistens Musikstücke, über den Computer an. Dafür wird der schon vorhandene Klang, die

Komposition, mit einem Gerät, dem Digitizer, in Computer-verständliche Form umgewandelt.

Im folgenden wollen wir uns den beiden ersten Methoden widmen. Die letzte, die sicherlich auch sehr interessant ist, behandeln wir hier nicht, weil sie im Hardware-Kapitel (es ist gleich das nächste) schon mit der entsprechenden Hardware angesprochen wird. Sie brauchen nur dort nachzuschlagen.

Die erste Gruppe der Sound-Software (welch ein Wort), wir wollen dies hier noch einmal zusammenfassen, nutzt also die altbekannte Notendarstellung, um Klänge - in diesem Fall ist es Musik - zu erzeugen. Wir haben hierbei drei wesentliche Elemente, die die Qualität eines solchen Programms ausmachen:

1. **Die Notation**

   Wichtig ist, wie die Noten dargestellt werden. Sind sie nur jeweils zeilenweise zu sehen, oder hat man den Überblick über ein ganzes System, wird in der Darstellung auch der Instrumententyp wiedergegeben, und werden keine Einschränkungen zum normalen Notenbild gemacht?

   Da wir es hier mit einem Editor zu tun haben, sollte uns nicht minder interessieren, wie man die Noten eingeben kann. Arbeitet das Programm mit der Maus und Symbolen zusammen, oder geht es lieber den Weg über die Menüs? Stellt es Funktionen zur Verfügung, mit denen man in der Lage ist, Teile des Stücks zu löschen, zu verschieben oder zu duplizieren? All dies sind Dinge, die später bei der Arbeit von entscheidender Bedeutung sein werden.

2. **Die Instrumentation**

   Neben der Notenausgabe haben wir es mit dem wichtigsten Punkt zu tun. Denn schließlich erzeugen die Instrumente den Klang, und somit sind sie verantwortlich für die Qualität des Programms. Hier ist es wichtig, daß neben fertigen Instrumenten, die auf der Diskette mitgeliefert werden, die Möglichkeit besteht, eigene Instrumente zu schaffen. Sei es durch das Einstellen oder Verändern von Parametern bestehender Instrumente oder durch das Laden digitaler Klangbilder von Instrumenten. Die letzte Methode stellt natürlich die absolute Spitze dar.

3. **Die Echt-Zeit-Eingabe**

   Entweder man begnügt sich mit den meistens mitgelieferten Musikstücken, oder man tippt welche ab. Diese sehr anstrengende und aufwendige Arbeit läßt sich meist dadurch abkürzen, daß die Programme eine sogenannte Echtzeit-Eingabe unterstützen, bei der die Tastatur des Amiga als Klaviatur mißbraucht wird. Damit lassen sich Improvisationen und schnelle Noteneingaben wundervoll gestalten und machen wesentlich weniger Arbeit.

Kommen wir jetzt endlich zur Software selbst. Als Vertreter für die Sound-Programme mit Notendarstellung ist Sonix von Aegis zu nennen. Dieses Programm ist nicht zuletzt wegen seines geringen Preises (er lag zur Zeit der Entstehung dieses Buches unter 150,- DM) sehr attraktiv.

Aegis Sonix

Das Programm präsentiert sich in einer hübsch gestalteten Verpackung: Ein fast DIN A4 großer Karton von etwa fünf Zentimeter Höhe mit farbenfrohem Aufdruck. Im Inneren finden wir ein ca. 250 Seiten dickes Handbuch, einige Werbeprospekte und die zwei Disketten, die allerdings in einer verschlossenen Tüte sind. Auf dieser ist eine Lizenzerklärung (natürlich in Englisch) aufgedruckt. Öffnet man die Verpackung, erkennt man, so Aegis, die Bedingungen an, die neben einigen anderen Klauseln das Kopieren der Software für nicht-privaten Gebrauch verbieten. Außerdem befindet sich weiterhin in der Tüte eine Postkarte, mit der man sich als Programmbesitzer registrieren lassen kann. Nur unter dieser Voraussetzung ist es möglich, ein späteres Update zu bekommen.

Die Anleitung

Einen besonders guten Eindruck macht das ausführliche Handbuch. Es ist in 13 Kapitel und 8 Anhänge geliedert. So beschreibt es zu Anfang erst einmal die Musik an sich. Dabei wird auf den Aspekt der Notierung genauso eingegangen wie auf die unterschiedliche Entwicklung der Musik zwischen West und Ost. Damit sind schon die ersten 50 Seiten gefüllt - und dies ist bestimmt nicht unnütz. Dann geht es an die Praxis! Zuerst wird das Starten des Pro-

gramms beschrieben. Hierbei geht man davon aus, daß die Workbench von der Sonix-Diskette geladen wird. Aber auch das Anklicken des Programm-Icons wird erwähnt. Viel später, nämlich erst im Anhang D, wird angesprochen, wie man ein BackUp von der Software macht. Gerade auf meinem Computer traten oft Ladefehler beim Orginal auf, weswegen dieser Punkt sehr wichtig ist.

In den folgenden Kapiteln wird Schritt für Schritt die Bedienung des Programms anhand von vielen kleinen Beispielen beschrieben. Erst gewinnt man einen groben Überblick, und nachher geht das Handbuch auch auf die Feinheiten ein. Somit ist der dritte große Teil für den aufsteigenden Anwender gedacht, der schon Erfahrung mit dem Programm hat. In den Anhängen finden wir Kurzinformationen zu den Menüs, den Kompositions-Techniken, den Tastatur-Kommandos und einigen anderen Dingen.

Das Programm

Zum Starten legen wir die Programm-Diskette oder eine Kopie in das Boot-Laufwerk. In ein vielleicht vorhandenes zweites Laufwerk legt man die zweite Diskette, auf der sich Musikstücke oder andere Daten befinden. Mindestanforderung für Sonix sind ein Amiga 500 mit 512 KByte, ein Farbmonitor und ein Kickstart größer oder gleich 1.2. Dieses letzte Kriterium erfüllen wir auf jeden Fall. Und den Farbmonitor kann man natürlich auch über Graustufung bei einfarbigen ersetzen, da die Farbwerte sehr kontrastreich sind. Viel Arbeit ist zu erwarten, wenn man mit nur einem Laufwerk gesegnet ist, weil Sonix die Instrumente auf der Programm-Diskette abgelegt hat.

Ist der Start erfolgreich verlaufen, finden wir das Anfangsbild vor uns: die Einstellungsseite für die Instrumente. Da wir damit aber überhaupt noch nichts anfangen können, wählen Sie bitte gleich im ersten Menü den Punkt Score aus. Wir haben jetzt das Notensystem in den oberen zwei Dritteln des Bildschirms - Sonix hat für sein Programm eine eigene Screen geöffnet - und im unteren Drittel die Noten und Einstellungen. Zu den Notensymbolen zählen alle Notenlängen, die Sonix zu spielen beherrscht, und die dazugehörigen Pausen, weiterhin noch Erhöhungs- und Erniedrigungszeichen. Wichtig zum Arbeiten sind die Editier-Funktion und das Instrumentenzeichen, mit dem ein Wechsel des Klanges (Instruments) herbeigeführt werden kann. Darunter fin-

den wir dann Schieberegler für Lautstärke, Spielgeschwindigkeit, Tonhöhe und Klangfarbe. Damit haben wir einen groben Überblick zum Programm.

Es gibt nun zwei Möglichkeiten, in der Arbeit mit dem Programm vorzugehen. Die erste und einfachste ist das Laden eines fertigen Musikstückes. Das haben wir auch zuerst getan. Hieran läßt sich schon einmal ausprobieren, wie weit die Einstellungen der Regler sich auf das Stück auswirken. Nun sollte man aber daran gehen, eigene Noten niederzuschreiben. Zuvor sehen Sie noch ein Bildschirmfoto, damit Sie sich ein Bild von der Notendarstellung machen können:

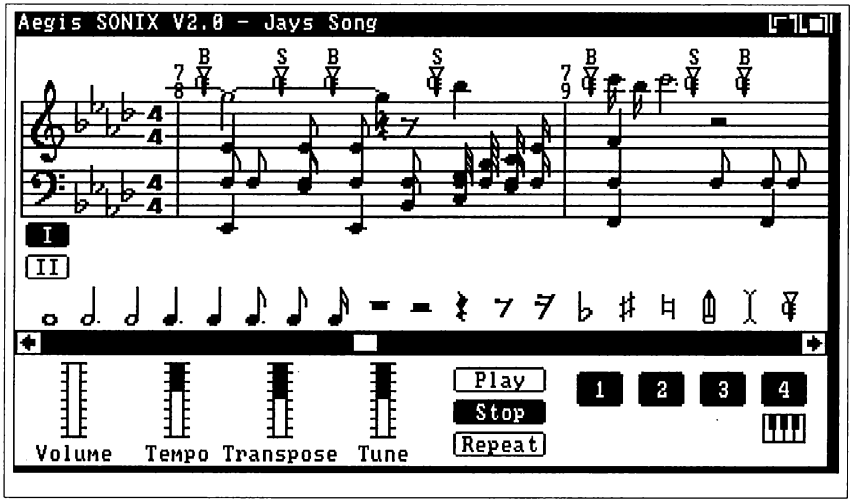


Abb. 3.10: Die Notendarstellung von Aegis Sonix

Dazu habe ich ein kurzes Lied aus meinen alten Noten ausgegraben und versucht, es einzugeben. Hierbei sollte man nicht der Illusion verfallen, in wenigen Minuten seien orchestrale Klänge hinzuzaubern. Am besten ist es, wenn man sich erst einmal mit der Melodiestimme begnügt. Die Eingabe stellt dabei keine Schwierigkeit dar. Alle nötigen Funktionen zum Editieren bestimmter Notenbereiche stehen zur Verfügung, und auch ausgefallenere Wünsche können erfüllt werden.

Der mit der Maus markierte Bereich kann entfernt und somit gelöscht werden. Sie können ihn aber auch an einer anderen Stelle wieder einfügen. Besonders

gut eignet sich die Kopier-Funktion, wenn Sie Stücke schreiben, die sich z.B. in der Begleitstimme wiederholen. Dann schreiben Sie diese Noten einmalig, markieren den Bereich und kopieren dies so oft, wie Sie wollen.

Aber auch ganz andere Funktionen werden zur Verfügung gestellt. Mit Play wird nur der markierte Bereich gespielt. Das macht es möglich, nicht immer das gesamte Musikstück zu spielen, was gerade bei längeren Werken viel Zeit spart. Um besonders kurze Abschnitte klanglich näher analysieren zu können, kann man auch eine Endlos-Spielfunktion anwählen, die den Namen Repeat trägt. Wieder anderer Natur sind die letzten vier Menüpunkte. Mit ihnen läßt sich der Block in seiner Tonlage verändern: Entweder mit Octave eine ganze Oktave oder mit Halfstep einen Halbtonschritt. Je nach Anwendung sind beide Möglichkeiten gefragt. Oktaven verlagern nur die Tonhöhe; mit den Halbtönen kann man aber den Charakter eines Liedes vollkommen verändern.

Genauso einfach wie das Notenschreiben läßt sich auch das Versehen mit Instrumenten durchführen. Sonix hält dafür ein Verzeichnis mit verschiedenen Klanggebern zur Verfügung. Sie wählen nur den Namen aus und können schon, mit dem Symbol für Instrumente, dieses an die gewünschte Stelle plazieren. Ab jetzt wird hier der Notentext vom angegebenen Instrument gespielt.

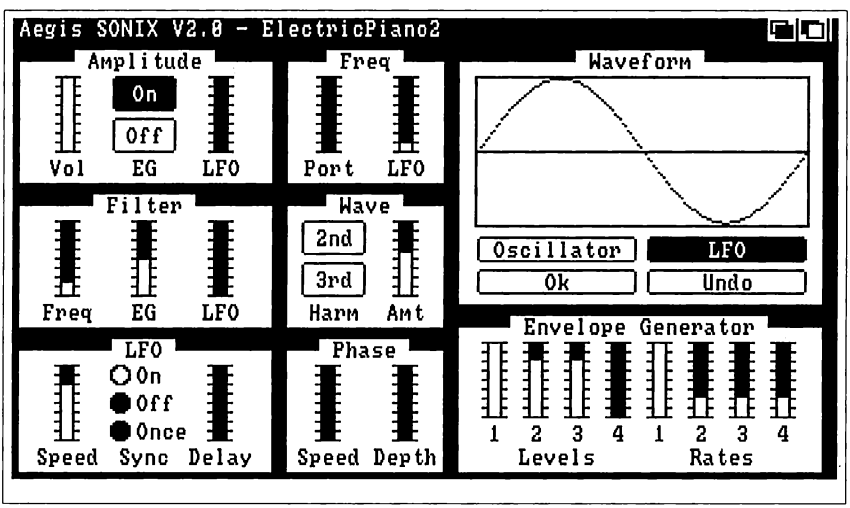


Abb. 3.11: Die Einstellungen aller Instrument-Parameter

Richtig interessant wird die Arbeit mit Sonix aber erst, wenn man beginnt, eigene Instrumente und Melodien zu kreieren. Dafür stellt das Programm einen separaten Programmteil ab, in dem man alle Parameter eines Klanges beeinflussen kann. Als wichtigstes seien hier nur die Wellenform und die Hüllkurve genannt. Die Wellenform kann innerhalb eines abgeteilten Bereiches mit dem Maus-Cursor grafisch beliebig eingestellt werden. So ist es auch möglich, Klänge zu konstruieren, die real nicht existieren können. Aber genauso werden auch die Standard-Wellenformen bereitgestellt wie etwa Sinus und Rechteck.

Noch viele andere Einstellungen kommen dazu wie Phasen, ein Amplituden-Modulator, Filter, mit denen das Klangvolumen verändert werden kann und ein Niedrig-Frequenz-Oszillator, so daß auch fortschreitende Klangmuster möglich sind. Diese selbstkomponierten Klangformen lassen sich dann auf der Diskette speichern, um für Musikstücke verwendet werden zu können. In Abb. 3.11 sehen Sie die Bildschirmseite, auf der sich alle Regler zur Klangbeeinflussung befinden.

Bis es aber soweit ist, empfiehlt es sich, ein neu geschaffenes Instrument erst einmal auszuprobieren. Was liegt da näher, als die Tastatur als Keyboard zu benutzen? Sonix hat deshalb in einem wiederum extra gelegenen Programmteil eine Screen parat, auf der man die Belegung der Tastatur mit Noten eingeben kann. Nach dem Start des Programms ist natürlich eine Belegung vorgegeben, aber Änderungen sind jederzeit möglich und erlaubt. Sie klicken dafür einfach auf das für die gewünschte Taste vorgesehene Feld, und dieses erscheint schwarz. Nun geht man mit der Maus auf die Klaviatur und klickt dort auf die Taste, die den Ton repräsentieren soll. Schon sind wir fertig! Ab jetzt liegt der ausgesuchte Ton auf der markierten Taste. Der Nachteil liegt nur darin, daß man als eingeübter Tastenspieler eines Instrumentes nur sehr wenig mit den [Amiga]-Tasten anfangen kann. Es ist leider nur wenig mehr als eine Spielerei, es reicht zum Ausprobieren eines neuen Klanges, weil die Tasten zu jeder Zeit des Programms aktiv sind, aber es kann auf keinen Fall die richtigen Tasten eines Klaviers oder Keyboards ersetzen.

DeluxeMusic Construction Set

Ähnlich wie das Zeichenprogramm aus der Deluxe-Serie, Deluxe Paint III, präsentiert sich auch das DeluxeMusic Construction Set. In einer aufwendigen

Schachtel etwa im Format DIN A4, die künstlerisch interessant aufgemacht ist, findet man, wenn auch etwas verloren, das Handbuch (leider nur im Format DIN A5), eine Registrierungs-Karte und die Diskette mit dem Programm.

Alles bekommt man für den stolzen Preis von 199,- DM incl. MwSt.

Der erste Eindruck

Wenn Sie sich von dem preislichen Schock etwas erholt haben, sollten Sie gleich wissen, daß dieses Programm auch seinen Preis wert ist. Wir werden uns aus diesem Grund vor dem Handbuch gleich einmal den Bildschirm ansehen und ein Musikstück laden.

Dann können Sie leicht erkennen, was dieses Programm ausmacht:

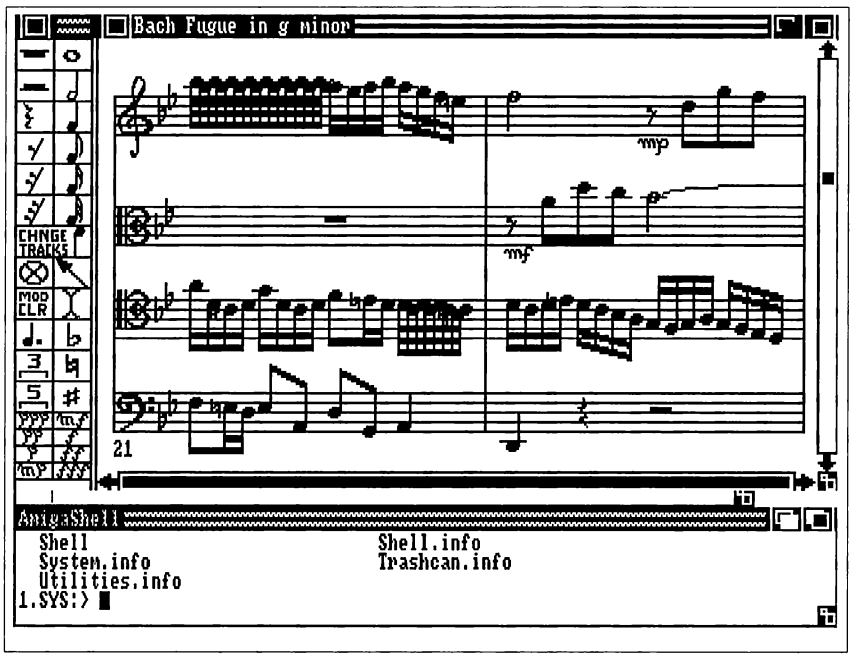


Abb. 3.12: Ein Musikstück mit dem DeluxeMusic Construction Set

Es läßt sich schon am Anfang eine gute Organisation erkennen. Ähnlich wie bei Sonix stehen alle Notensymbole zum Anklicken mit der Maus auf der linken Seite in einem Window zur Verfügung. Besser und aufwendiger ist aber die Notendarstellung an sich gelöst. Wir haben das gesamte Notensystem auf dem Bildschirm in einem Window. Neben dem Standard-Notenschlüssel kann auch der Bratschenschlüssel ausgewählt werden. Insgesamt ist das Notenbild etwas naturgetreuer als bei Sonix, doch darauf werden wir an anderer Stelle noch kommen.

Das Handbuch

Die etwa 100 Seiten starke Anleitung ist im üblichen Deluxe-Stil von Electronic Arts aufgemacht. Nach einer kurzen Beschreibung beider Autoren folgt das Inhaltsverzeichnis. Schon hier wird an der kleinen Schrift nicht nur klar, daß dieses Programm nur für Nicht-Brillenträger gedacht ist, sondern auch, daß die vielen Informationen sehr gedrängt untergebracht sind. Mit der inhaltlichen Unterteilung wurde es wie immer gehalten: zuerst haben wir einen Überblick über das vollständige Programm, in dem gezeigt wird, wie man ein Musikstück aus den beigefügten Demos lädt, dieses startet, etwas verändert und dann abspeichert. Der Benutzer ist damit gleich mit dem Programm in Kontakt gekommen und lernt seine Bedienung kennen. Schon in diesem Teil finden wir viele kleine Bildschirmausschnitte wieder, die es dem Anfänger leichter machen, sich auf dem Bildschirm zurechtzufinden.

Das zweite Kapitel geht nun alle Funktionen, die das Programm anbietet, Schritt für Schritt durch. Hier macht es sich nun bemerkbar, daß dieses Handbuch in englischer Sprache geschrieben ist. Uns lag zum Test leider nur diese Version vor. Wir können deshalb die Aussagen nur unter Vorbehalt treffen, weil eine deutsche Anleitung wesentlich zum Verständnis beigetragen hätte. Da helfen auch die Erklärungen im Anhang der Anleitung nur sehr wenig weiter. Interessant war aber, daß, obwohl es sich um eine PAL-Version handelte (es soll also die gesamte Zeilenzahl von 256 ausgenutzt werden) das Deluxe Music Construction Set nicht dazu zu bewegen war, dies auch voll auszunutzen, weil die Screen ganz einfach nur mit 200 Zeilen geöffnet wurde.

Dem etwas versierteren Anwender tut das dritte Kapitel gute Dienste. In ihm sind noch einmal alle Funktionen des Programms mit Kurzbeschreibungen (der Namen hält, was er verspricht) versehen. An manchen Stellen haben wir

hier auch eine Schritt-Für-Schritt-Anleitung gegeben. Dies hilft in Situationen, bei denen man sich nicht mehr genau sicher war, was zuerst zu tun ist. Zu den schwierigeren Operationen, wie z.B. der Endbearbeitung des Notenbildes, findet man auch Ratschläge, die alle wichtigen Bereiche abdecken.

Im Anhang findet man als suchender Leser nicht nur die Beschreibung des Sonderzeichensatzes für die Notengrafik und das Stichwortverzeichnis, sondern auch eine Beschreibung, wie man das Programm zusammen mit einer Midi-Schnittstelle benutzt. Anhand einer Ablaufbeschreibung und einem kleinen Beispiel müßte es jedem, der über die nötige Hardware verfügt, möglich sein, auch dies zu realisieren.

Das Handbuch läßt sich, abgesehen von dem Manko, daß es in Englisch verfaßt ist, mit einer guten Note beurteilen. Neben dem kurzen Einsteigerteil, der dem "heißen" Anwender die ersten Erfolgserlebnisse vermittelt, können sich das zweite und dritte Kapitel wirklich sehen lassen, weil in ihnen jede vom Programm unterstützte Kleinigkeit erwähnt und mit Beispielen beschrieben wird.

Die Programmbedienung

Wie man an dem Musik-Beispiel gesehen hat, ist die Darstellung der Noten sehr an die allgemein üblichen Konventionen angelehnt. Es stellt sich für uns jetzt die Frage, wie sich das Programm bedienen läßt, wenn man es dazu benutzt, eigene Stücke einzugeben. Dafür gehen wir zuerst einmal davon aus, daß ein fertiges Stück abgetippt werden soll.

Die Grundeinstellungen wie Tonart, Notenschlüssel und Takt lassen sich problemlos in Requestern, die über die Menüs angewählt werden, einstellen. Beides kann natürlich auch noch innerhalb des Stückes gewechselt werden. Nun geht es aber an die Eingabe. Dafür wählt man sich aus dem Noten-Menü, das sich in dem schmalen Window links befindet, die entsprechende Note aus, setzt sie an die richtige Stelle auf die Notenlinien, und schon ist die erste Note geschrieben. So verfährt man mit dem gesamten Musikstück. Allerdings sollte man sich dabei nicht vorstellen, daß alles wie von selbst geht. Beide Programme - ob man sich nun Sonix oder DMCS ansieht - erfordern einen sehr hohen Arbeitsaufwand, bis eine kleine Melodie eingegeben ist.

Der entscheidende Unterschied liegt beim Deluxe Music Construction Set aber darin, daß die Noten nach der Eingabe noch wesentlich in ihrer Darstellung verbessert werden können. Die Fähnchen können über Balken verbunden werden, gebundene Noten können mit Bögen als solche gekennzeichnet werden und auch die anderen Feinheiten sind zugelassen. So gibt es Triolen, Crescendos, Wiederholungen und Anschlag-Einstellungen. Damit ist das Grundwerkzeug für eine ausgefallene Komposition vorhanden.

Neben der Vollständigkeit aller musikalischen Mittel sind aber auch die Bearbeitungs-Werkzeuge interessant. Hier bietet DMCS genau wie Sonix die Standard-Funktionen. Es können Bereiche markiert werden. Diesen Bereich kann man dann duplizieren und an anderer Stelle wieder einfügen. Er kann aber auch gelöscht oder verschoben werden. Auch sämtliche Einstellungen beziehen sich in den meisten Fällen auf den markierten Bereich.

Besondere Funktionen des Programms

Neben den Standard-Funktionen, die man schon als üblich betrachten kann, besitzt das Deluxe Music Construction Set aber auch andere, die weniger dem Standard angehören. Sie sollen hier punktuell angesprochen werden.

So gibt es beim DMCS einen Requester zur Einstellung der Signatur, den man sowohl für die Grundeinstellungen als auch für die Wechsel während des Stückes verwendet. Hierbei gibt es drei wesentliche Einstellungen, die diesen Requester von anderen abheben. Er bietet nämlich neben allen Tonarten noch die Funktion an, den vielleicht schon existierenden Notenbereich zu transponieren. Damit ist es für den Komponisten oder Solisten eine Kleinigkeit, Musikstücke, die nicht in der benötigten Tonart geschrieben sind, in die gewünschte zu übertragen.

Ein anderer Punkt sind die Symbole für bestimmte Spielarten. So benötigen einige Musikinstrumente, wie z.B. die Gitarre einige Sonderzeichen, die bestimmt nicht von jedem Programm unterstützt werden können. Auch DMCS kann an dieser Stelle nicht weiterhelfen. Da aber am Ende der Kette immer der wirkliche Musiker steht, der das Stück spielen soll, reicht es in diesem Fall, wenn man den Notentext mit den nötigen Zeichen versieht, die dann ja auch auf dem Ausdruck wiedergegeben werden.

Mit diesen Zeichensätzen ist es natürlich auch jederzeit möglich, Kommentare in den Notentext zu setzen oder diesen gar mit Texten für Gesang zu versehen. Sie haben dafür eine komplette Texteinbindung zur Verfügung.

Der Gesamteindruck

Das Deluxe Music Construction Set zeichnet sich durch seine nahe Anlehnung an die wirkliche Komposition aus. Es gibt die Noten in einer üblichen Darstellung aus, unterstützt so gut wie alle Feinheiten der musikalischen Differenzierung und bietet hervorragende Bearbeitungseigenschaften. Diesen letzten Punkt erfüllt auch Sonix, aber man vermißt bei diesem Musikprogramm die Details. Viele Kleinigkeiten, die eine Arbeit komfortabel gestalten, sind leider nicht vorhanden.

Was Sonix aber auszeichnet, ist der Instrument-Editor, den man bei DMCS leider vergeblich sucht. Wer darauf mehr Wert legt, hat einen Grund, sich dieses Programm zuzulegen. Wer aber mehr auf die spätere Anwendbarkeit in der "realen" Musik achtet, ist mit DMCS sehr gut bedient.

Audiomaster II

Neben der Grafik ist der Amiga auch ein sehr leistungsfähiger Computer auf dem Gebiet der Ton- und Klangerzeugung. Großen Anreitz bietet dabei immer noch die Möglichkeit, bestehende Töne und Klänge festzuhalten, zu digitalisieren und im Computer zu bearbeiten. Das hier getestete Programm bietet diese Möglichkeiten unter einer neuen Bedieneroberfläche. Es sind neue Funktionen unter der Version 2.0 zusammengekommen, die noch mehr Auswahl und Vielfalt bieten sollen.

Ist man stolzer Besitzer des Audiomaster II (für knapp DM 150,-) geworden, findet man vor sich eine fast DIN A4 große Schachtel, in der sich Handbuch, eine Registrierkarte und ein Umschlag mit zwei Disketten befinden. Die Disketten sind wie auch bei Sonix in diesem Umschlag, da man vor dem Öffnen noch eine Copyright-Erklärung absegnen muß. Diese garantiert ungeschützte, also kopierbare Software, bedeutet aber auch eine gewisse Hürde. Hat man nun aber doch die innere Schranke überwunden (schließlich möchte man das Programm ja nutzen), liegen zwei Diskette vor einem. Auf der einen Diskette

befindet sich das (auch bootfähige) Programm mit der Workbench 1.3 und auf der anderen einige Beispiele zum Anhören und weiterverarbeiten.

Das Handbuch umfaßt etwa 80 Seiten und erklärt in 10 unterschiedlich langen Kapiteln die Handhabung des Programms. Es ist in englischer Sprache gehalten, wie auch alle Funktionen innerhalb des Programms mit englischen Texten versehen sind. Dies dürfte aber, zumindest bei der Programmbedienung, keine Schwierigkeiten darstellen, da uns solche Begriffe wie "Volume", "Digital" oder "Sound" von der Stereo-Anlage hinreichend vertraut sein müßten. Und auch die Anleitungstexte sind schon mit einigem Schulenglisch leicht zu verstehen und beschreiben in kleinen Schritten die Programmanwendungen.

Dieses Musikprogramm hebt sich von den bisher vorgestellten (Sonix, DMCS) ab. Ziel dieses Programms ist es nämlich nicht, wie bei den anderen beiden Programmen, Noten einzugeben und abzuspielen, sondern Töne über Mikrophon oder Stereoanlage einzulesen und weiterzuverarbeiten. Wir haben es mit einem Sample-Programm zu tun, mit dem wir wie in einem Tonstudio Klänge erzeugen, aufzeichnen und verfremden können.

Nach dem Start des Programms, ob über Autoboot oder Doppelklick im Disk-Fenster, meldet es sich mit folgendem Bildschirm, der einen Überblick über die Arbeitsweise bietet (wir haben hier schon eine Wellenform geladen, damit Sie gleich mit der Anzeige vertraut werden):

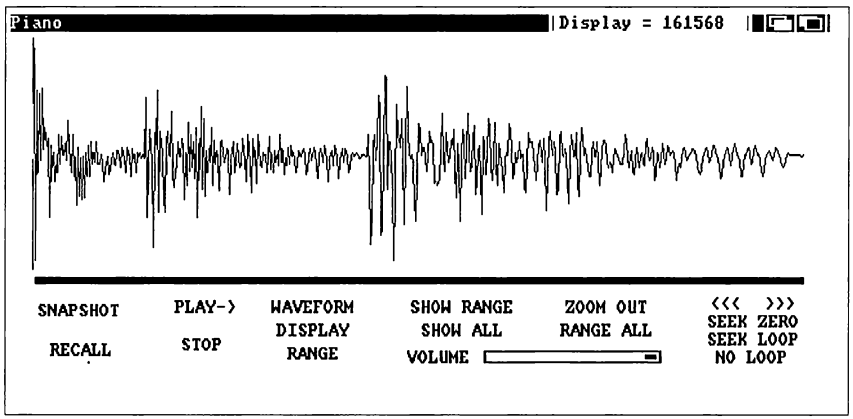


Abb. 3.13: AudiomasterII

Die obere Hälfte wird für die Darstellung der Klangformen benutzt. Dabei kann zwischen Stereo- und Mono-Betrieb gewählt werden. Standardmäßig ist Stereo eingestellt, was man an der Unterteilung in zwei Oszillogramme sieht (so wird die grafische Darstellung von Schwingungen bezeichnet).

Die untere Hälfte bietet vier Gruppen von Gadgets, die zum schnellen Bearbeiten der Klangformen benutzt werden. Allerdings sind dies natürlich nicht alle Funktionen. Ergänzt wird die Funktionspalette noch um weitere sechs Menüs, in denen die elementaren Steuerungen untergebracht sind. Neben den Diskettenfunktionen finden sich hier zwei Editier-Menüs, eines für Spezialeffekte bzw. die benötigten persönlichen Einstellungen und ein Menü, in dem die HiFi-Tonausgabe unterstützt wird.

Zum Test des Programms haben wir mit einem preiswerten Digitizer verschiedene Klänge digitalisiert. Zwar ist Audiomaster stereofähig, doch dies zwingt nicht zur Verwendung eines Stereo-Digitizers. Nur bei Musikstücken ist dies empfehlenswert. Doch damit sind wir gleich bei einem besonderen Problem. Allgemein kann man sagen, daß die Einspeisung von Musikstücken Probleme mit sich bringt. Dies liegt vor allem an den hohen Aufzeichnungsfrequenzen, die für einen klaren Klang nötig sind. Hierfür ist zum ersten ein sehr guter Digitizer mit einer hohen Digitalisierungsrate notwendig. Außerdem hat auch die Software ihre Grenzen, die es natürlich nicht möglich machen, CD-Qualität zu erreichen.

Unser Rat ist deshalb: Beschränken Sie sich auf die Digitalisierung von Klängen, die man in der Umwelt findet. Und auch Sprache eignet sich relativ gut. Setzt man diese Fragmente später mit Hilfe der Stereo-Funktion zusammen, lassen sich äußerst interessante Effekte erzielen. Leider können wir hier keine Klangprobe geben.

Kommen wir jetzt zu den Funktionen, die uns das Programm zur Verfügung stellt. Hat man erst einmal einen Klang in den Computer eingelesen (man kann für ganz Eilige auch einen von der Diskette laden), so geht es an die Veränderungen. Dargestellt wird die gesamte Wellenform. Dies kann bei sehr langen Digitalisierungen irritierend wirken, da bei der Darstellung von 50 Sekunden keine Datails mehr zu erkennen sind. Dafür gibt es die Möglichkeit, Ausschnitte zu bearbeiten und diese vergrößert darzustellen.

Innerhalb dieser Ausschnitte oder auch des gesamten Sounds können nun die Beeinflussungen einsetzen. Neben dem Herausschneiden von Klängen oder

dem Einfügen vollkommen anderer Klangstücke sind solche Dinge wie Echos oder Lautstärkeregelungen sehr effektvoll. Hierbei läßt sich nicht einfach nur ein Echo erzeugen. Das Echo selbst kann in wichtigen Einstellungen beeinflußt werden. Neben der Anzahl der Echos und dem Zeitintervall, in dem sie sich wiederholen sollen, kann auch der Lautstärkeabfall in einem sehr feinen Bereich eingestellt werden.

Die so modifizierte Wellenform kann dann erneut vom Programm abgespielt werden. Nützlich ist dabei die Funktion zum Abspielen eines markierten Bereiches, wobei man viel Zeit spart, wenn man nur die Veränderung hören möchte.

- Abspielen in verschiedenen Variationen
- Wie man editiert (Schnippseltechnik)

Software-Überblick:

Programm:	Audiomaster - Sound-Digitizer
Preis:	DM 150,-
Version:	V II 1.00
Lieferumfang:	Handbuch mit 80 Seiten, 2 Disketten
Hardware:	mind. 512 KByte und Sound-Digitizer
Kickstart:	ab 1.2

3.5 Desk Top Publishing

Die Anwendungsmöglichkeiten unseres Amiga steigern sich immer mehr. Nicht nur im Bereich der Text- und Datenverwaltung wird er für "seriöse" Anwendungen genutzt. Durch die Grafikfähigkeit bietet er sich immer deutlicher auch in kreativen Gebieten an. Allerdings sind damit nicht immer nur die Mal- und Zeichenprogramme gemeint. Schließlich gibt es auch hier wesentlich mehr Anwendungen.

So bieten zwar viele Textverarbeitungs-Systeme Gestaltungsfunktionen an, die man früher nicht von ihnen erwarten konnte - die Einbindung von Grafi-

ken gehört heute schon zu einem Standard, doch für sogenannte "Desk Top Publishing"-Anwendungen reicht es bei dieser Programmgattung dann doch nicht.

Was ist eigentlich "Desk Top Publishing"? Übersetzt heißt dieser Begriff etwa soviel wie "Vom Schreibtisch aus puplizieren". Damit ist die fertige Erstellung von Dokumenten gemeint, die zu Informationszwecken dienen sollen. Vorstellbar sind Werbeseiten, interne Informationsblätter und sogar Zeitungen. Alles dies faßt man unter dem Begriff "Desk Top Publishing" zusammen.

Benötigt werden für das "Desk Top Publishing" (wir wollen es im folgenden Text nur noch kurz mit DTP bezeichnen) nicht nur die Möglichkeit, Texte zu verarbeiten, denn dann wäre kein Unterschied zu einer normalen Schreibmaschine zu sehen, sondern auch mächtige Funktionen zur grafischen Gestaltung. Dazu gehört einmal die Formatierung des Textes in professioneller Art, wie man es von Zeitschriften gewohnt ist, und zum zweiten zählt man dazu auch das Einbinden von Grafiken in geeigneter Weise.

Wie dies genau auszusehen hat, werden wir am Beispiel von CityDesk zeigen.

City Desk

Das Programm wird mit einem Ringordner als Handbuch geliefert und ist zum jetzigen Zeitpunkt zum Preis von 198,- DM zu erhalten. Zu Testzwecken lag uns bereits eine eingedeutschte Version vor, die aber alle Umlaute verarbeitet. Es traten nur einige Schwierigkeiten bei der Übernahme von Fremdformaten auf. Mit diesem Programmpaket bekommt man neben der schon beschriebenen Text-Formatierung auch einen Grafik-Editor und eine Seiten-Gestaltung vorgelegt.

Für dieses Programm benötigen Sie mindestens 512 KByte RAM. Es hat sich aber gezeigt, daß eine Speichererweiterung von ganz besonderem Vorteil ist. Denn sonst erhalten Sie schon bei kleineren Dokumenten die Meldung "Kein Speicher mehr vorhanden". Dies liegt ganz besonders daran, daß neben dem Text sehr viel Grafik verwaltet werden muß. Diese kann aber bei 512 KByte nur wenig Platz im Chip-RAM finden, da dort ja schon das Programm und andere systeminterne Daten liegen. Deshalb ist Fast-RAM ein Muß.

Genauso sollte man bei DTP auch gleichzeitig an einen verbesserten Monitor denken, denn die beste Bearbeitung läßt sich in der Screen-Auflösung von 640x512 Punkten erreichen. Doch hier macht der Interlace-Modus den Augen ganz schön zu schaffen. Es gibt deshalb geeignete Hardware wie Bildschirmvorsätze, die dieses Flimmern lindern. Jedoch ist ein spezieller Monitor oder eine Hardware-Erweiterung natürlich weit effektiver.

Haben Sie City Desk gestartet, öffnet es eine eigene Screen (leider wird die Workbench nicht geschlossen, wodurch Grafikspeicher gespart werden könnte). Auf dieser sieht man ein Fenster vor sich, in dem die erste Seite mit zwei leeren Textspalten dargestellt wird. Nun ist es am Anwender, diese Seite zu füllen. Dies kann mit Texten oder mit Grafiken geschehen, die von anderen Programmen importiert werden. Für diese Arbeit sollte man sich schon im klaren sein, wie die spätere Seite aussehen soll. Denn im Gegensatz zu anderen DTP-Programmen, die auch um einiges teurer sind, beinhaltet City Desk keinen Text-Editor.

Vor einer Seitengestaltung liegt noch eine grundsätzliche Formatierung, die nichts mit dem Text oder der Grafik im Detail zu tun hat, sondern sich auf die Formatierung der Seite als ganzes bezieht. Wie viele Spalten Text, welche Randbreite oder das Papierformat an sich sind die Fragen, die zuerst beantwortet werden müssen!

Sie sollten also zuerst mit einer Textverarbeitung, dies kann auch das Notepad oder der CLI-Editor ED sein, alle Texte vorbereiten. Sie können aber auch während der Arbeit einen Text auf der Workbench-Screen neu entwerfen oder verändern. Innerhalb von City Desk ruft man den vorher erarbeiteten Text über "Text einlesen" auf. Mit der Maus kann dieser Text im Standard-Zeichensatz erst einmal positioniert werden. Alle weiteren Einstellungen folgen.

Dafür sind im einzelnen die Auswahl des Zeichensatzes, die Schriftattribute und die Ausrichtung zu bestimmen. Als Zeichensätze werden alle auf der Workbench vorhandenen und die zusätzlich mitgelieferten Laser-Fonts akzeptiert. Letztere nur bei einem Laser-Drucker, wie es der Name schon sagt. Dazu kommen dann die Einstellungen wie Fettdruck, Kursivschrift oder Unterstreichen. Genauso lassen sich Absätze im Blocksatz, zentriert oder entweder rechts oder links ausrichten.

Der Text wird vom Programm dann je nach Schriftgröße neu in die Spalten der Seite eingefügt. Diese Funktion wird besonders interessant, wenn man be-

ginnt, Grafiken in den Text einzufügen. Dies können alle im IFF-Format abgelegten Bilder sein, die mit bis zu 32 Farben ausgestattet sind. Da City Desk aber nur Schwarz-Weiß-Grafik verwaltet, werden beim Einlesen des Bildes die Farben in Graustufen umgewandelt.

Diese Grafik wird genau wie auch der Text nach dem Einlesen mit der Maus positioniert und ist ab sofort in die Seite integriert. Gegebenenfalls vorhandener Text, der sich mit der Grafik überschneidet, kann um diese herumfließen.

Dazu ruft man die Funktion "Textkettenfluß" auf und läßt damit die Verteilung erneut berechnen.

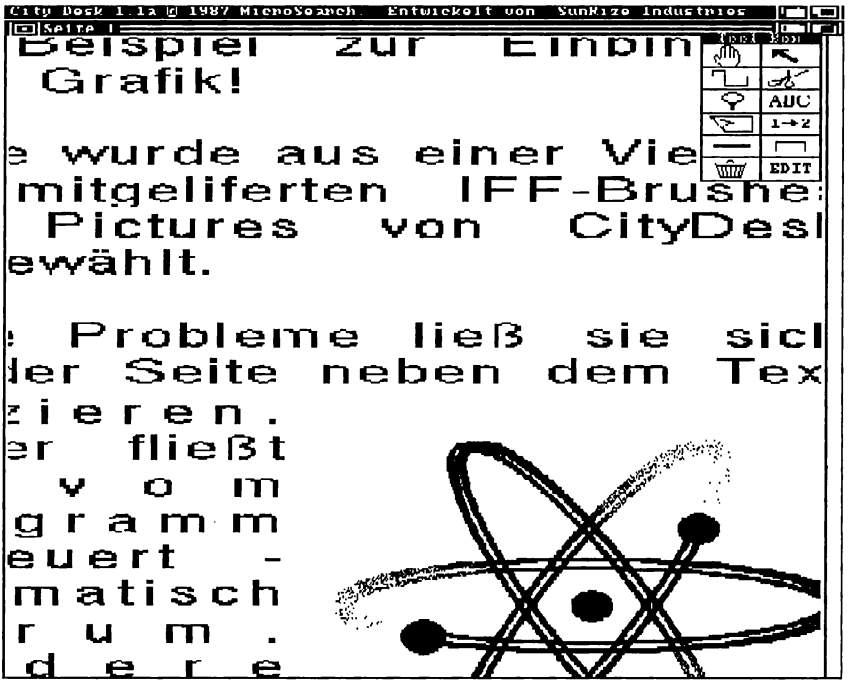


Abb. 3.14: City Desk

Die Gestaltung einer Seite kann somit ihren Lauf nehmen. Für die weitere Verarbeitung bietet City Desk einige hilfreiche Funktionen an. So können Texte, die von dem DTP-Programm als Ketten verwaltet werden, getrennt oder wieder zusammengefügt werden, wenn es der Textfluß fordert. Ein Edi-

tieren der Texte ist zwar mit Hilfe eines String-Gadgets noch möglich, verlangt aber große Ausdauer vom Anwender. Hier liegt ein entscheidender Nachteil des Programms.

Großzügiger ist da der Grafik-Editor. Es gibt einen speziellen Screen, der alle Grund-Zeichenfunktionen wie Linien, Kreise und Rechtecke in einem Menü enthält und es erlaubt, mit Ausschnitten zu arbeiten. Dadurch können aus ganzseitigen Grafiken Teile herausgeschnitten werden, die sich besser in den Fließtext einarbeiten lassen. Die Größe der Grafik oder des Ausschnitts selbst ist dabei immer noch variabel. Genauso die Position, die jederzeit bis auf den Punkt genau (hier im wahrsten Sinne des Wortes) bestimmt werden kann.

Als abschließendes Beispiel sehen Sie in Abb. 3.14 eine mit City Desk erstellte Seite. Wir haben als Schriftart einen großen Zeichensatz gewählt, damit bei der Verkleinerung einer DIN-A4-Seite auf DIN A5 (unser Buchformat) noch etwas zu erkennen ist.

Zur DTP-Software von City Desk gehört übrigens nicht nur das DTP-Programm, sondern auch ein ganz besonderes zusätzliches Tool. Da bestimmt viele Anwender unter chronischem Speicher-Mangel leiden, wurde das Programm CD Print mitgeliefert, das es ermöglicht, Dokumente oder Seiten bestehender Dokumente ohne den Editor auszudrucken. Dadurch wird viel Speicher gespart, und auch aufwendigere Seiten sind ohne Probleme auszudrucken.

Software-Überblick:

Programm:	City Desk - Desk Top Publishing
Preis:	DM 198,-
Version:	1.1a
Lieferumfang:	Handbuch mit etwa 100 Seiten, 2 Disketten
Hardware:	mind. 512 KByte (je mehr desto besser)
Kickstart:	ab 1.2

Superbase

Ein weiterer Vertreter der Datenverwaltungen ist Superbase. Dieses Programm ist genau wie DATAMAT in der Lage, beliebige Datenstrukturen nach dem Karteikastenprinzip zu verwalten. Interessant wird auch dieses Programm letztendlich durch die Verknüpfungsmöglichkeiten, die es gestatten, mit hoher Geschwindigkeit auf bestimmte Einträge zuzugreifen.

Das Programm kostet in der kleinen Version ca. 200,- DM und bietet insgesamt zwei weitere Aufbaustufen. Die Professional Version für knapp 400,- DM und die Entwickler-Version für gute 600,- DM. Unser Test soll sich hauptsächlich auf die kleinere Version beziehen, da diese schon alle wichtigen für den normalen Anwender nötigen Funktionen enthält. Die bei größeren Versionen vorhandenen Ergänzungen lassen sich nur in Spezialfällen nutzen oder bieten gestalterische Ergänzungen und sind für uns deshalb weniger von Interesse.

Auffällig ist der etwas unbequeme Kopierschutz. Das Programm an sich läßt sich ohne Probleme kopieren und ggf. auch auf Festplatte installieren. Jedoch ist ein sog. Dongel nötig, um es auch starten zu können. Unter einem Dongel versteht man einen Stecker, der an einen Joystickport gesteckt wird. In dem Stecker befindet sich eine Schaltung, die von der Software abgefragt wird. Das Programm arbeitet nur, wenn dieser Stecker am Computer betrieben wird.

Nach dem Starten des Programms wird ein großes Window auf der Workbench-Screen geöffnet. Standardmäßig benutzt Superbase diese Ausgabeart. Ganz unten findet man ein separates Window, in dem lauter Symbole untergebracht sind, die dem Bedienungsfeld eines Video-Rekorders ähnlich sind. Mit diesen wird man später die Steuerung durch die Dateien aufnehmen können.

Will man nun eine Datei laden, wählt man dazu "Datei-->Öffnen" im Projekt-Menü und erhält die File-Request-Box, in der alle Dateien des eingestellten Datei-Verzeichnisses aufgeführt sind. Nach der Auswahl wird die Datei geladen. In den folgenden Abbildung sehen Sie eine Datei, wie Sie im Window als Texttabelle angezeigt wird:

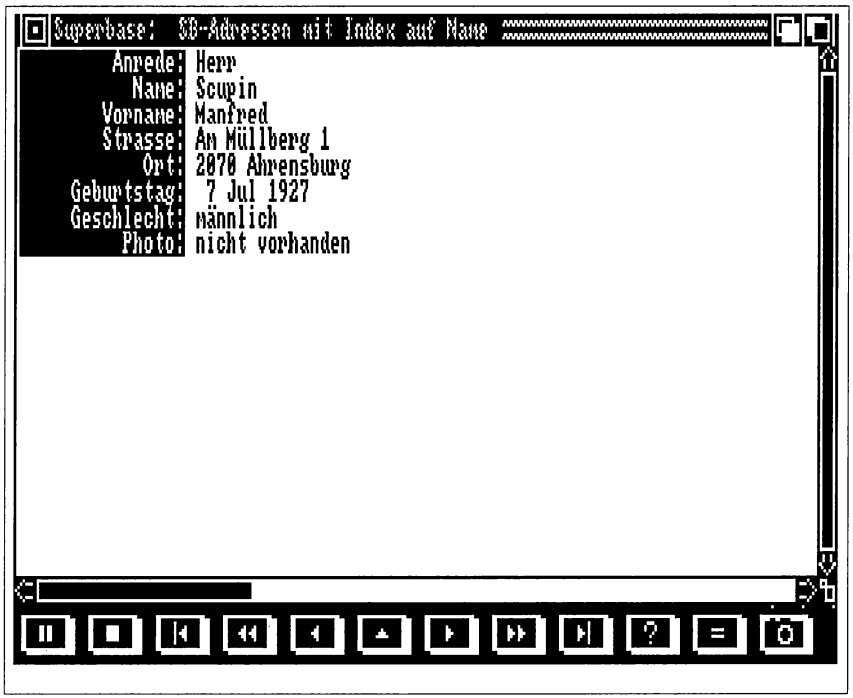


Abb.3.15: Superbase-Dat

Ähnlich wie im Standard-Modus des DATAMAT werden alle Datenfelder untereinander angezeigt. Mit einem Mausklick kann man dann bequem in eines der Felder gehen und dort den Eintrag editieren. Über die Cursortasten oder mit der [Return]-Taste gelangt man in die anderen Felder.

Es geht natürlich auch anders. Die doch recht spartanische Anzeige erinnert an die alten Zeiten, in denen noch nicht auf ansprechende grafische Gestaltung Wert gelegt wurde. Wie man es machen kann, zeigt eine Abbildung aus der Professional-Version, die Sie am Ende des Berichtes sehen können.

Dabei ist es möglich, die gesamte Formularseite wie eine Grafik zu verwalten und damit auch zu gestalten.

Für unseren Test der Datenverwaltung soll auch hier eine Probedatei erstellt werden. An diesem Vorgang kann man am besten überprüfen, ob das Pro-

gramm hält, was es verspricht. Dazu wird wie auch bei vorhergehenden Tests die Datei "Test-Adressen" erstellt, in der alle wichtigen Datentypen vorkommen.

Man wählt also den Menüpunkt "Projekt-Neu-Datei" und wird sogleich nach deren Namen gefragt. Weiterhin kann schon hier die Vorgabe eines Paßwortes (versionsbedingt) erfolgen. Dieses ist im Unterschied zum DATAMAT nur ganzheitlich zu geben und kann nicht eingeschränkt genutzt werden, so daß z.B. ein Mitarbeiter oder ein Bekannter zwar Daten sichten aber nicht verändern kann.

Danach erfolgt die Eingabe der einzelnen Feldnamen. Jeweils dazu passend kann man mit Gadgets den Typ des Feldes auswählen. Als Feldtypen stehen Text, Numerisch und Datum bzw. Zeitangaben zur Verfügung. Ergänzt werden die Parametertypen durch Kennzeichnungen wie Extern, Erwartet, Nur Lesen, Überprüfung, Berechnung und Konstante, die mit den vom DATAMAT bekannten Einstellungen vergleichbar sind.

Nach dem Abschluß der Dateidefinition befindet man sich in der Eingabe. Hier kann in gewohnter Weise die Dateneingabe erfolgen. Hat man mehrere Datensätze eingetragen, kommen die Symbole im unteren Window zur Geltung. Wie bei einem Tonbandgerät oder einem Videorekorder können Vorspulen, Pause, Start oder Ende eingestellt werden. Analog dazu werden die Datensätze vorgespult, das Vorspulen hält an, der erste oder der letzte Datensatz wird angezeigt. Die Handhabung ist einfach und unkompliziert.

Das erste Problem bei der Handhabung ist aufgetreten, als mit mehreren Dateien gearbeitet wurde. Superbase hat grundsätzlich nichts gegen das Arbeiten mit mehr als einer Datei. Allerdings kann immer nur eine Datei zur Zeit im Ausgabe-Window angezeigt werden. Nur durch erneutes Öffnen gelangt man in eine andere Datei.

Ganz besonders glänzt diese Dateiverwaltung in der Flexibilität der Dateienmasken. So kann nach dem Öffnen einer Datei zu jedem Feld beliebig ein Index erstellt werden. Dies geschieht ebenfalls über die Funktion "Öffnen". Weiterhin stehen dabei eine Feldauswahl und eine Abfrage zur Verfügung. Mit Hilfe der Feldauswahl können nur bestimmte Felder der Datei auf dem Bildschirm ausgegeben werden. Damit lassen sich leichter Daten eingeben, wenn alle übrigen Felder unberücksichtigt bleiben sollen. Diese Selektion in der Datenerfassung und Betrachtung ist mit der Möglichkeit bei DATAMAT

zu vergleichen, verschiedene Masken für die Bildschirmausgabe zu definieren. Unter dem Begriff Abfrage, der auch in diesem Menü zu finden ist, verbirgt sich eine sehr mächtige Möglichkeit, mehrere Dateien zu kombinieren und deren Daten zu Listen oder neuen Dateien zusammenzufassen.

Dieser Prozeß läuft im einzelnen so ab, daß man in einem Requester neben der Titelzeile alle Felder angeben kann, die in der Zielliste bzw. der Zieldatei ausgegeben werden sollen. Dazu kommt eine Report-Funktion, die es erlaubt, zwei Felder zu verknüpfen und damit ein neues Datum zu schaffen.

Mit der Filter-Option ist es zusätzlich möglich, nur bestimmte Datensätze in diese Verarbeitung einzubeziehen. Letztendlich kann diese durch die drei vorangegangenen Funktionen definierte Datensatzliste noch nach einer Vorschrift sortiert werden.

Es läßt sich also sagen, daß die Funktionen der Superbase-Software in ihrer ganz eigenen Weise mächtige und in ihrer Anwendung sehr weitreichende Funktionen zur Datenverarbeitung bereitstellen.

Zum Abschluß möchten wir Ihnen noch zeigen, welche Gestaltungsmöglichkeiten durch den Masken-Editor in der Professional-Version zu erreichen sind. Die mitgelieferte Datei "Superbase People" enthält aber nicht nur die trockenen Daten, sondern kann auch mit Signet und Konterfeit angeschaut werden.

Bevor Sie nun vor der Entscheidung stehen, entweder Superbase oder DATAMAT für ihre persönlichen Daten heranzuziehen, sollten Sie genau auf die Art achten, mit der ihre Datenverarbeitung arbeitet.

Legen Sie mehr Wert auf eine grafisch ansprechende Datensatzdarstellung, liegt DATAMAT schon in der Grundversion besser als Superbase. Möchten Sie aber alleine Daten verwalten und legen wenig Wert auf die Darstellung, so kann auch Superbase überzeugen.

Ein klares Plus für Superbase sind die schon in der Grundversion vorhandenen Verknüpfungsfunktionen, die bei DATAMAT erst in der Professional-Version durch die eigene Programmiersprache erreicht werden. Ansonsten bleibt aber immer noch der Weg zu Fuß durch die Selektions- und Sortier-Funktionen.

Software-Überblick:

Programm:	Superbase - Datenverwaltung
Preis:	DM 199,-/399,-/599,- je nach Version
Version:	2.03
Lieferumfang:	Handbuch ca. 250, 2 Disketten
Hardware:	512 KByte
Kickstart:	ab 1.2

3.6 Utilities

TURBOprint

Das Festhalten von Daten auf Papier ist schon immer eine der größten Bestrebungen des Menschen gewesen. So ist es auch beim Amiga nicht weiter verwunderlich, daß der Anwender seine Texte und Grafiken auf Papier bringen will.

Um größtmöglichen Bedienkomfort zu garantieren, hatte man sich bei der Entwicklung des Amiga etwas besonderes einfallen lassen. Es wurden Druckertreiber in das System integriert, die einen perfekten Ausdruck gleich welchen Druckers gewährleisten sollten, ohne daß sich das betreffende Programm darauf einzustellen hat.

Hierbei ergaben sich aber einige Probleme. Zum ersten haben diese Druckertreiber durch ihre (angeblich) hohe Flexibilität eine große Geschwindigkeitseinbuße gebracht, die sich besonders bei Grafiken bemerkbar macht. Und zum zweiten sind die Druckertreiber meistens nur in der Lage, das, was auf dem Bildschirm ist, in sehr einfacher Weise darzustellen. Was fehlt, ist die Möglichkeit, der Druckbeeinflussung, um schönere und grafisch bessere Ergebnisse, besonders bei Bildschirmkopien, zu erzielen.

Dafür gibt es ein Programm, das die mitgelieferten Druckertreiber und die Einstellungs-Software (Preferences) ergänzen bzw. ersetzen soll. TURBOprint stellt diesen Anspruch und ist für ca. DM 90,- zu haben. Es wird zusammen mit einem etwa 30 Seiten umfassenden Handbuch geliefert, in dem alle Fragen zur Installation und Programm-Bedienung erklärt werden. Außerdem befindet sich auf der mitgelieferten Diskette das Programm, das leider mit ei-

nem Kopierschutz versehen wurde. Dadurch muß man bei der Arbeit mit einem Backup immer wieder die Orginal-Diskette einlegen. Dies wird besonders dann störend, wenn man TURBOprint auf die Festplatte überträgt.

Bei der täglichen Arbeit mit TURBOprint lernt man als Anwender besonders schnell, die Vorteile zu nutzen. Das größte Manko, die unzureichende Unterstützung der Druckervielfalt hat jetzt endlich ein Ende. Mitgeliefert werden insgesamt 81 Druckertreiber, die nicht nur alle wichtigen Drucker-Fabrikate wie PANASONIC, OKIDATA, NEC, CANON, HP, SEIKOSHA, STAR und EPSON unterstützen (so auch die NEC-Drucker mit ihren Fähigkeiten), sondern auch eine Auswahl der Grafikauflösung ermöglichen. So muß nicht immer in der gröbsten Auflösung "genadelt" werden und auch nicht immer im Schönschreibdruck, der viel Zeit kostet.

Es gibt eine residente HardCopy-Funktion, die es jederzeit erlaubt, die im Vordergrund befindliche Screen auszudrucken. Diese Grafik kann aber auch im IFF-Format - also ladbar und verständlich für jedes Grafikprogramm - auf einen angegebenen Datenträger geschrieben werden. Dafür eignet sich besonders die RAM-Disk oder aber eine vielleicht vorhandene Festplatte.

Im Gegensatz zu den Standard-Preferences läßt sich TURBOprint selbst auch resident machen. Dadurch können Sie bestimmte Einstellungen viel schneller wechseln, als es bisher möglich war. Das Wichtigste ist aber der Geschwindigkeitsvorteil, der ausschlaggebend für die Entscheidung zu diesem Programm ist. Zum Test wurde ein sehr bekanntes Bild ausgewählt: der Affenkopf. Dieses unter Public Domain zu erhaltene Bild sollte nun unter den verschieden Bedingungen zu Papier gebracht werden; dafür wurde ein NEC Pinwriter P6+ verwendet, der mit seinen 24 Nadeln eine gute Druckqualität aufweist. Es ergaben sich bei unseren Tests folgende Druckzeiten:

Unter der normalen Workbench:

```
1.2 WB: 11 Minuten (Druckertreiber CBM_MPS2xxx)
1.3 WB:  4 Minuten (Druckertreiber Nec_Pinwriter)
```

Unter der Workbench mit TURBOprint

```
1.2 TP:  1 Minute  (Druckertreiber NEC/Px_090x180dpi)
1.3 TP:  1 Minute  (s.o.)
```

An diesem Beispiel zeigt sich neben der enormen Geschwindigkeitssteigerung innerhalb der Workbench-Versionen, zu welchen Verbesserungen TURBOprint trotzdem noch in der Lage ist.

Das Programm zeichnet sich aber nicht alleine durch Geschwindigkeit aus.

Ein Hauptargument für den Kauf sollte auch die Viefalt der Einstellungen zur Grafikausgabe sein. Sehen Sie hier die Abbildung der Hauptseite, die nach dem Start des Programms erscheint:

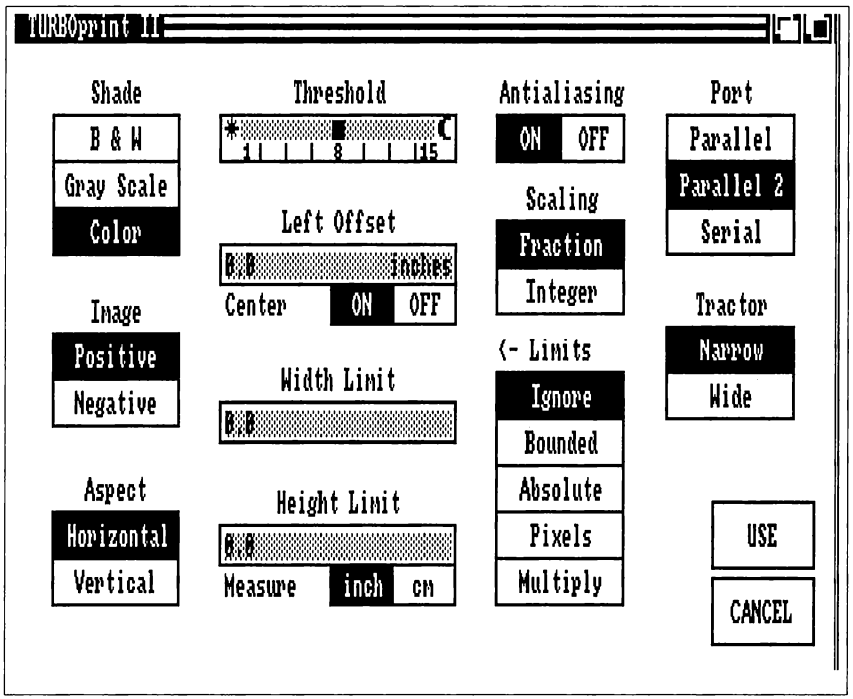


Abb. 3.16: TURBOprint-Hauptseite

Die Bedienung des Programms ist stark an die Software Preferences angelegt, so daß der Benutzer wenig Zeit zur Umstellung benötigt. Im einzelnen gibt es hier die Druckerauswahl auf der linken Seite unten. Dabei können über das Hauptverzeichnis das Drucker-Fabrikat und im Unterverzeichnis noch die Druckqualität ausgewählt werden.

Darüber finden wir noch eine Einstellung zur Umsetzung der Farb- oder Graustufenschattierung in bestimmten Mustern. TURBOprint bietet dafür insgesamt 6 Varianten.

Neben den Einstellungen zum Seitenverhältnis oder der Umsetzung von Helligkeit und Kontrast findet man auch die Möglichkeit einer Hardcopy-Option, die das Bild entweder zum Drucker schickt oder im eingestellten Verzeichnis abspeichert.

Interessant ist das "Untermenü" CallPrefs, in dem detailliert Druckparameter eingestellt werden können. Die Einstellungen gleichen sehr stark denen des Preferences-Programms von der Workbench 1.3.

Sehen Sie dazu das Window, das nach dem Aufruf erscheint:

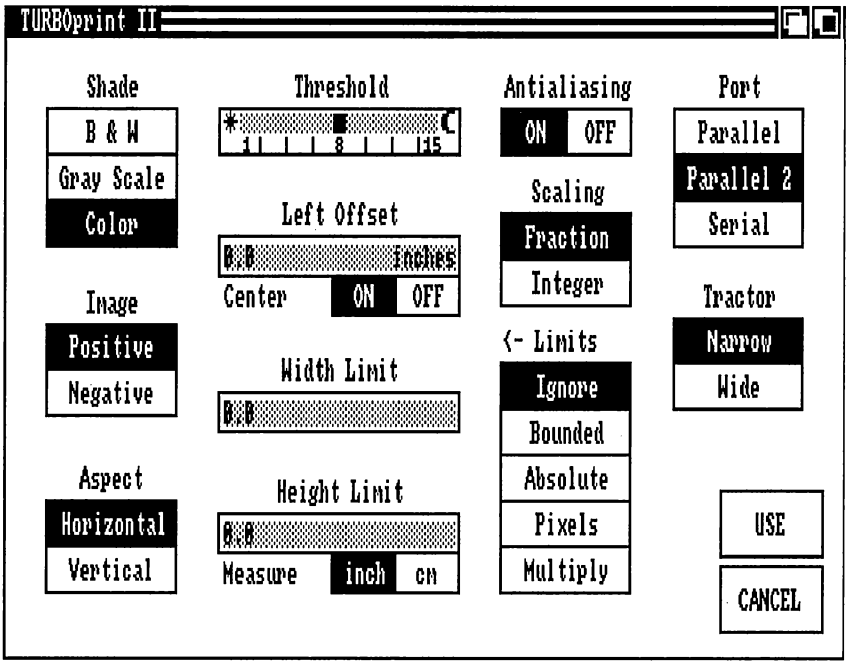


Abb 3.17: TURBOprint-Parameter

Vertraut sind die Limit-Einstellungen, das Scaling und das Antialiasing aller weiteren Gadgets bilden eine Zusammenfassung der Einstellungen aus den allgemeinen Printer-Einstellungen der Workbench. Die Geschwindigkeit im Ausdruck wird in jedem Fall verbessert, und so macht sich das Programm bei einer Vielzahl von Ausdrucken auf jeden Fall bezahlt.

Software-Überblick:

Programm:	TURBOprint - Ausdruckbeschleuniger
Preis:	DM 99,-
Version:	2.01
Lieferumfang:	Handbuch 34, 1 Diskette
Hardware:	512 KByte
Kickstart:	ab 1.2

DISCovery

Für die tägliche Arbeit ist ein Diskettenmonitor zwar nicht nötig, doch schon kleinere Defekte an wichtigen Datendisketten, ein wenig Lust zur Spielerei oder der Reiz der Gefahr bringen einen dazu, einen Diskettenmonitor zu gebrauchen. Natürlich lassen sich damit auch gelöschte Daten wiederherstellen bzw. versteckte erzeugen.

DISCovery ist ein solcher Diskettenmonitor, mit dem es möglich ist, die Daten auf einer Diskette näher zu betrachten (das Wort Monitor soll für Disketten-Untersucher stehen und nicht für Bildschirm!).

Das Programm wird für ca. DM 198,- angeboten. Geliefert wird es mit einem Handbuch in einem Format zwischen DIN A5 und DIN A4 von etwa 160 Seiten Stärke. Beigefügt waren bei unserem Testexemplar 2 identische Disketten. Nach Angaben des Herstellers enthält das Programm keinen Kopierschutz, doch wie soll man die Abfrage nach Textstellen aus dem Handbuch sonst bezeichnen. Allerdings bietet sich so die Möglichkeit, ein Disketten-Backup oder eine Kopie auf der Festplatte anzufertigen. Trotzdem war das Nachschlagen im Handbuch später so lästig, daß wir uns eine Liste der ge-

fragten Worte neben meinen Computer gelegt haben. Nach der Abfrage eines Wortes aus dem Handbuch bietet sich dem Anwender folgendes Bild:

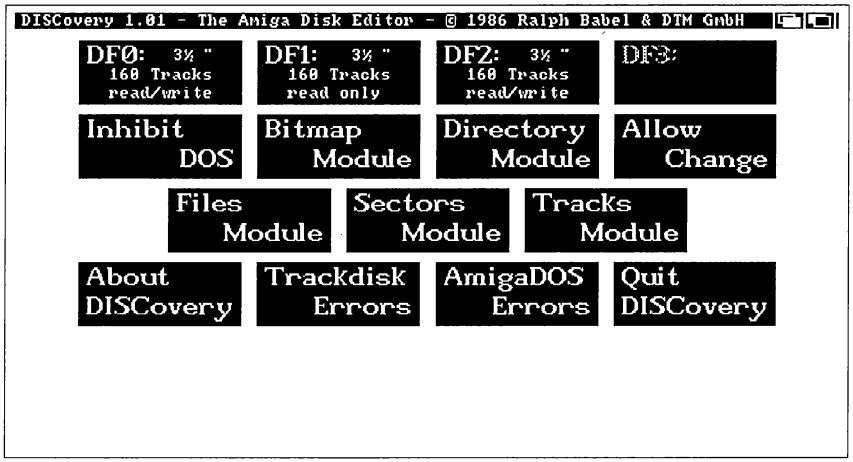


Abb. 3.18: DISCovery "Main-Screen"

Wie man sieht, bietet das Programm insgesamt sieben Programmfunktionen: Das Bitmap-Module, das Directory-Module, das Files-Module, das Sectors-Module und das Tracks-Module. Außerdem gibt es noch zwei Fehler-Untersuchungs-Routinen Trackdisk-Errors und AmigaDOS-Errors. Hierbei fällt gleich auf, daß das gesamte Programm in Englischer Sprache ist. Und dies, obwohl es ein deutsches Produkt ist. Ein trauriger Umstand, wo es doch bei der Flexibilität des Amiga kein Problem wäre, die Sprache beliebig zu ändern!

Weiterhin kann in der obersten Zeile eines der vier möglichen und auch angeschlossenen Diskettenlaufwerke ausgewählt werden. Zusätzlich stehen zwei Einstellungen zur Verfügung. Mit der ersten "Inhibit DOS" wird der AmigaDOS-Zugriff auf das behandelte Laufwerk unterbunden. Dies verhindert z.B. lästige Requester, die auf eine zerstörte Diskettenstruktur hinweisen. Der zweite Schalter stellt eine Überprüfung des Programms dar, die verhindert, daß man während seiner Arbeit die Diskette wechselt. Im Normalfall verbietet dies DISCovery. Es ist aber durch Einschalten dieses Gadgets trotzdem möglich. Alle Programmaufrufe, die hier beschrieben wurden, sind auch in den drei Menüs, die das Programm anbietet, vorhanden. Begibt man sich in eines der Module, wechselt die Menüleiste. Als Beispiel soll uns dazu das Direc-

tory-Module dienen. Nach dem Aufruf dieses Programm-Moduls lädt DISCovery zuerst den gesamten Verzeichnis-Baum des eingestellten Diskettenlaufwerks (das kann schon einige Zeit dauern). Die danach angezeigte Tabelle enthält alle Informationen, wie man sie auch mit List aus dem AmigaDOS aufrufen kann. In den Menüs werden nun alle nötigen Funktionen angeboten, um die vorliegenden Einträge zu modifizieren. Neben Name, Protection-Bits (alle auch der Version 1.3), Zeit und dem Filenote-Eintrag kann zusätzlich das erste Daten-Byte eines Files rekonstruiert werden. Dies ermöglicht die Reparatur eines File-Header-Blocks, was sonst nicht möglich wäre.

Damit sind die Möglichkeiten aber noch nicht erschöpft. Es lassen sich z.B. Einträge neu erstellen oder bestehende löschen. Mit diesem Möglichkeiten kann man schon einiges anstellen. So z.B. Einträge im Verzeichnis schaffen, die gar nicht existieren. Oder wiederum andere, die es vorher gab, löschen.

Wesentlich globalere Informationen über die Diskettenstruktur bietet das BitMap-Modul. In ihm findet man die Belegung der gesamten Diskette wieder. Jeweils durch ein Symbol wird in der Tabelle der Zustand jedes Sektors beschrieben. Es läßt sich so sehr leicht feststellen, welche Datenblöcke für Verwaltung, welche für Organisation und welche nicht benutzt sind.

Für die genauere Betrachtung eines Sektors, sei es ein Daten- oder ein Organisations-Sektor, gibt es das Sektors-Module. In diesem Unterprogrammteil werden die genauen Daten sowohl in ASCII als auch in (hexa-)dezimaler Schreibweise aufgelistet. Mit dem Cursor lassen sich so ganz leicht z.B. Texte auf der Diskette verändern oder ergänzen. Mit Hilfe der Menüfunktionen lassen sich diese Daten verändern. Es können neue Hashs oder die Prüfsumme berechnet werden, genauso wie man untereinander verbundene Datenblöcke wie an einer Perlenschnur verfolgen kann. Zusätzlich zu den Daten an sich in jedem Sektor listet das Programm an der Seite die jeweiligen Datenblock-Typen auf, so daß der Anwender einen schnellen Überblick über Organisationsblöcke gewinnen kann.

Im Files-Module, das ähnlich wie im Sektors-Module mit den Daten auf der Diskette arbeitet, wird die Sektor-Einteilung aufgehoben. Die gesamte Datenmenge eines Files steht dem Anwender zur Verfügung und kann nach Herzenslust oder mit Bedacht korrigiert oder einfach nur verändert werden. Man muß diese Anwendung aber eher zur Betrachtung rechnen, da sich in der Praxis weit häufiger Probleme mit Disketten ergeben, die man möglichst ohne Datenverlust lösen möchte.

Für diese Probleme bietet der Diskettenmonitor zwei recht leistungsfähigke Module, mit denen es möglich ist, in die Struktur einer Diskette einzugreifen. Dazu zählen das Sectors- und das Tracks-Module. In Verbindung mit der Fehleranalyse Trackdisk-Errors und AmigaDOS-Errors können so durch Schreibfehler entstandene Datenverluste in den meisten Fällen durch Wiederherstellung ausgeglichen werden.

Dazu kann im Tracks-Module mit der Funktion "read defective tracks" eine Korrektur vorgenommen werden, die auf einer neuen Diskette abgespeichert werden kann. Durch Kopieren der noch funktionsfähigen Datensegmente läßt sich so eine neue Diskette zusammenbasteln, die keine Fehler mehr enthält.

Abschließend läßt sich zu dem Programm sagen, daß es viele äußerst interessante Informationen über eine Diskette aufzeigt. Die mitgelieferte Anleitung gibt neben einer gut und ausführlich erklärten Anwendungsbeschreibung auch Hintergrundeinblicke in die Diskettenorganisation, so daß man als Laie noch etwas lernen kann.

Als einziger Nachteil sollte bemerkt werden, daß die Menüführung vollkommen in englischer Sprache ist. Dies versperrt dem unkundigen Anwender den Zugang, da man sich neben den vielen Informationen über den Diskettenaufbau auch noch mit den englischen Fachbegriffen rumschlagen muß. Dies wird aber von der Möglichkeit aufgewogen, jederzeit die Anzeigeformate in den drei wichtigen Zahlensystemen (hexadezimal, binär, dezimal) auszuwählen.

Software-Überblick:

Programm:	DISCovery - Diskettenmonitor
Preis:	DM 198,-
Version:	1.01
Lieferumfang:	Handbuch 163, 2 Disketten
Hardware:	512 KByte
Kickstart:	ab 1.2

4. Hardware

So, nachdem Sie mittlerweile ein richtiger Software-Profi geworden sind, können wir uns dem nächsten großen Kapitel zuwenden. Es geht um Hardware. Unter Hardware versteht man all das, was sichtbar, faßbar und nicht ohne weiteres kopierbar ist. Also Drucker, Laufwerke, die Maus, den Monitor, den Amiga selbst und natürlich all die vielen kleinen undurchschaubaren Dinge, aus denen Computer und deren Zubehör zusammengesetzt sind.

"Ohne Hardware nützt die beste Software nichts", ist eine Weisheit, die über die zwingende Notwendigkeit, einen Computer zu besitzen, auf dem die Software läuft bzw. funktioniert, hinausgeht. Es gibt nämlich Programme, die nicht auf jedem Amiga arbeiten bzw. auf einem Gerät gut und auf einem anderen nur schlecht oder gar nicht laufen. Dabei handelt es sich nicht etwa um die Bevorzugung einiger Amiga-Besitzer, sondern um unterschiedliche Hardware.

Falsch ist jedoch, daß es Amiga-Programme gibt, die z.B. nur auf einem Amiga 2000 laufen. Der Schein trügt hier nämlich. Solche Programme funktionieren auf einem Amiga 500 in der Grundausstattung nämlich nur deshalb nicht, weil dem 500er die nötigen Extras fehlen. Rüstet man diese Extras nach, laufen auch die Programme. Doch welche Extras gibt es? Dieser Frage wollen wir hier nun auf den Grund gehen.

Vorweg jedoch schon mal ein kleiner Ausblick auf das, was Sie noch erwartet:

Zunächst beschäftigen wir uns mit den verschiedenen Speichererweiterungen. Der 500er kann relativ leicht und billig um 512 KByte auf 1 MByte aufgerüstet werden. Dabei kommt er meistens noch in den Genuß einer akku-gepufferten Echtzeituhr. Dieser Begriff wird Ihnen wahrscheinlich noch unbekannt sein. Haben Sie jedoch das Speicherkapitel gelesen, sind Sie ein richtiger Uhrenexperte.

Nachdem der Amiga einen größeren Arbeitsspeicher erhalten hat, wenden wir uns einer anderen Art der Speichererweiterung zu. Wir wollen uns dann mit dem Anschluß eines weiteren Diskettenlaufwerks befassen. "Nicht nötig", sa-

gen die einen, "Zu teuer", die anderen, und eine dritte Gruppe, die einmal mit einem Amiga mit zwei oder mehr Laufwerken gearbeitet hat, kann gar nicht genug Floppies (= Laufwerke) bekommen. Auch Sie werden spätestens dann, wenn Sie einige Wochen mit dem Amiga gearbeitet haben, das Bedürfnis nach einem zusätzlichen Laufwerk verspüren.

Festplatten kommen als Speichermedien auch nicht zu kurz. Sie sind zwar zur Zeit noch etwas teuer, werden jedoch rasch billiger werden. Hier verraten wir schon, daß eine Festplatte ein riesiges Laufwerk ist, dessen "Diskette" nicht gewechselt werden kann. Festplatten sind dafür aber unerhört schnell.

Auch mit wenig Geld kommt man zu guter Hardware. Anhand eines Sound-Digitalisierers werden wir Ihnen das beweisen. Für Bastler haben wir auch einige Leckerbissen auf Lager. So können Sie z.B. die Geschwindigkeit Ihres 500ers steigern, indem Sie einen anderen Prozessor einsetzen. Das ist bei weitem nicht so schwierig, wie es sich anhört.

Möchten Sie mit Kickstart 1.1 oder 1.3 arbeiten? Kein Problem nach dem Einbau einer Kickstart-Umschaltung. Ärgern Sie sich darüber, daß Spiele und Programme nach dem Einbau einer Speichererweiterung nicht mehr funktionieren? Nichts leichter als das. Wir zeigen Ihnen, wie Sie diesen Fehler mittels eines kleinen Schalters beheben.

Sicher brennen Sie jetzt schon darauf, näheres zu den oben aufgeführten Themen zu erfahren. Wenn dies so ist, lesen Sie doch einfach weiter. Wir starten mit den Speichererweiterungen.

4.1 Speichererweiterungen

Der Amiga 500 wird in der Grundausstattung mit einer Speicherkapazität von 512 KByte geliefert. 512 KByte sind 512 KiloByte, ein KiloByte sind wiederum 1024 Bytes und ein Byte entspricht acht Bit. Ein Bit ist die kleinst mögliche Speichereinheit. Es kann nur zwei Zustände haben: 0 oder 1 bzw. Strom oder kein Strom. Ein aus acht Bits bestehendes Byte kann nun nicht acht oder neun, sondern 256 verschiedene Zahlen darstellen. Diesen Wert erhält man aus der Rechnung 2 hoch 8. Zwei Möglichkeiten für jedes Bit und 8 Bits macht 256. Ein Byte kann auch in Zahlen kodierte Buchstaben und Sonderzeichen enthalten. Hat man z.B. 10 Bytes kann man darin zehn Buch-

staben ablegen. Der Amiga hat nun aber 512 mal 1024 Bytes. Dies sind 524288 Bytes. Sie können also 524288 Buchstaben im Speicher ablegen.

Um zu verdeutlichen, wieviel das ist, nehmen wir einmal an, Sie wollen einen 512 KByte langen Text eingeben. Weiterhin erreichen Sie 200 Anschläge pro Minute und vertippen sich nicht. Sie müßten ca. 43 Stunden tippen, tippen und nochmals tippen. Ins Buch der Rekorde kämen Sie aber erst dann, wenn Sie länger als 222 Stunden an einem Stück tippen würden. Doch das nur am Rande.

Führt man die Speicherkapazität auf diese Weise vor, kommt man meistens zu dem Schluß, daß man mit dem vorhandenen Speicher sehr gut auskommen müßte. Wenn man jedoch bedenkt, daß eine Grafik leicht 80000 Bytes lang sein kann, wird man von diesem Standpunkt schnell wieder abrücken. Überlegt man weiterhin, daß professionelle Programme oft eine Länge von über 150000 Bytes aufweisen, wird man einsehen, daß man mit 512 KByte schnell in Bedrängnis geraten kann.

Nebenbei erkennt man übrigens auch, wieviel Arbeit in solchen Programmen steckt. Zur Erstellung eines Programms sind oft etliche Millionen Tasten zu drücken. Sollten Sie also beabsichtigen, längere Programme zu schreiben, können Sie sich mit 512 KByte gleich begraben lassen. Für kleine und mittlere Programme (besonders für solche, die in BASIC geschrieben sind) reicht der vorhandene Speicher jedoch meistens vollkommen aus. Auch C-Compiler kann man mit 512 KByte gerade noch betreiben. Hier ist 1 MByte jedoch wärmstens zu empfehlen. Ganz gut kommt man mit den Compilern auch mit zwei Laufwerken und 512 KByte zurecht, wenngleich man bei dieser Lösung etwas Geduld aufbringen muß.

Erhalten Sie öfters Meldungen, denen zufolge der Speicher nicht ausreicht, ist es jedenfalls höchste Zeit, diesen aufzustocken. Dabei gibt es mehrere Möglichkeiten, die in unterschiedlichen Preisklassen anzusiedeln sind.

4.1.1 Erweiterung auf 1 MByte

MByte. Diese Abkürzung wurde schon mehrfach verwendet. Allen, die deren Bedeutung noch nicht kennen, sei gesagt, daß ein MByte eine andere Be-

zeichnung für 1024 KByte ist. Man kann sich also leicht ausrechnen, daß ein MByte 1048576 Bytes enthält. Eine ganz stattliche Zahl.

Unter der Bezeichnung A501 bietet Commodore eine Karte an, die ohne Garantieverlust in den Amiga 500 eingebaut werden kann, und ihm zusätzlich 512 KByte Speicher verschafft, womit die Gesamtkapazität des Speichers auf 1024 KByte = 1 MByte wächst. Zusätzlich befindet sich auf dieser Karte eine akku-gepufferte Echtzeituhr. Diese Uhr kann, wenn Sie vorhanden ist, vom Amiga ausgelesen werden. Die Zeit, die diese Uhr anzeigt (oder sagen wir besser enthält, denn sehen kann man sie ja nicht), überträgt er dann in seine eigene Uhr. Und wozu das Ganze? Nun, die Uhr auf der Erweiterungskarte funktioniert auch dann noch, wenn Sie den Amiga ausschalten. Sie bezieht ihre Betriebsspannung aus einem kleinen Akku, der sich immer dann automatisch auflädt, wenn der Amiga eingeschaltet ist. Er kann die Uhr über eine Zeitspanne von einigen Monaten versorgen.

Diese Uhr ist besonders nützlich, wenn Sie Texte oder Programme schreiben, da diese immer eine Zeitangabe erhalten, die über den List-Befehl des CLI angezeigt werden kann. Sie können so immer leicht feststellen, wo die neuste Version Ihrer Datei zu finden ist. Doch kommen wir zurück zur Karte. Diese Karte ist zu einem Preis von ca. 300 DM über den Einzelhandel zu beziehen. Sie ist robust gebaut, verfügt jedoch über keinen Schalter, um den zusätzlichen Speicher abzuschalten. Legt man Wert auf eine abschaltbare Karte, kann man sich an die Produkte der Hardware-Hersteller halten. Die Firma Message bietet z.B. eine abschaltbare Karte an, die vom Anweder jedoch selbst mit RAM bestückt werden muß. Ebenfalls von dieser Firma wird ein ca. 30 Mark teurer Zwischenstecker angeboten, der es ermöglicht, eine nicht abschaltbare Speicherkarte (z.B. Commodore A 501) schaltbar zu machen.

Das Fehlen dieses Schalters ist nicht so unwichtig, wie man meinen könnte. Legen Sie jedoch Wert darauf, unbedingt die Karte von Commodore zu kaufen, können Sie dieses Problem umgehen, indem Sie NoFastMem von der Workbench aufrufen, das den neuen Speicher scheinbar abschaltet. Ein weiterer Aufruf aktiviert ihn wieder. Einige Programme funktionieren auch nach NoFastMem nicht. Aber auch dieser hartnäckigen Software - meist handelt es sich um Spiele - kann man vorgaukeln, daß nur 512 KByte Speicher zur Verfügung stehen. Dazu kann man das "neue" NoFastMem oder den NoFastMem-BootBlock verwenden. Beide sind in Kapitel 2.9 abgedruckt. Möchten Sie trotzdem eine Hardware-Lösung, so können Sie in Kapitel 4.6.7 nachlesen, wie nachträglich ein Schalter zu installieren ist.

Nun wissen Sie schon allerhand über die Erweiterungskarte. Verschwiegen haben wir bis jetzt noch, wo diese Karte eigentlich angeschlossen wird. Bevor wir jedoch auf den Einbau zu sprechen kommen, sollen noch die Fremdhersteller von Speicherkarten zu Wort kommen. Sie bieten Karten, die den Speicher um 512 KByte erweitern meist billiger an. Beim Kauf einer Fremdherstellerkarte sollten Sie unbedingt auf den Schalter und auf die Uhr achten. Wichtig ist auch, daß die Karte bestückt ist. Eine Hardware-Firma, die wahrscheinlich lieber nicht genannt werden möchte, und deshalb nicht namentlich erwähnt wird, bietet z.B. eine Speichererweiterung zum sagenhaft günstigen Preis von ca. 69,- DM an. Auf der Abbildung zeigt die Vertriebsfirma auch eine ganz normale Karte. Kleingedruckt gibt sie jedoch an, daß die Platine ohne RAM, also ohne Speicher geliefert wird. Sie würden also nicht die im Katalog abgebildete Karte mit 512 KByte, sondern eine Karte mit ein paar Widerständen und 0 KByte (in Worten NULL) Speicher kaufen! Die RAM-Bausteine müssen Sie sich also selbst besorgen, was Zeit und Geld kostet. Doch nicht nur das - Sie müssen die Chips auch einlöten - und das ist alles andere als einfach. Die Firmen sollten solche Mißverständlichkeiten übrigens vermeiden!

Wie gesagt, lassen Sie sich keine Karte ohne Schalter, Uhr und RAM-Chips andrehen!

Nach dieser Warnung können wir nun endlich den Einbau vornehmen. Dazu müssen Sie das Gerät abschalten. Sichern Sie vorher jedoch alle Daten. Am leichtesten können Sie den Einbau vornehmen, wenn Sie auch alle Stecker abschrauben und abziehen.

Nun drehen Sie den Amiga um. Auf der Unterseite können Sie einen Aufkleber mit der Seriennummer und eine Klappe sehen, die den Eindruck erweckt, für die Aufnahme der Speicherkarte vorgesehen zu sein. Dies ist auch richtig. Übrigens müßten Sie noch einen ovalen Aufkleber sehen, der als Garantiesiegel fungiert. Dieses Siegel interessiert uns an dieser Stelle aber noch nicht. Wir werden es zu einem späteren Zeitpunkt entfernen.

Vorerst gilt unsere ganze Aufmerksamkeit noch der Klappe. Sie zu öffnen kann schon das erste Problem darstellen. Drehen Sie den Amiga so, daß der Teil der Unterseite, in dem sich die Klappe befindet, nach links zeigt. Die Ports (z.B. RGB, Maus und Strom sollten nun nicht sichtbar von Ihnen wegzeigen. In dieser Stellung legen Sie den 500er auf eine Decke oder einen Tisch.

Nun nehmen Sie einen Schraubenzieher mittlerer Größe zur Hand, den Sie in den länglichen Schlitz der Klappe schieben. Dieser Schlitz befindet sich, wenn Sie den Amiga so hingelegt haben, wie von uns vorgeschlagen, ganz links. Nun schieben Sie den Griff des Schraubenziehers vorsichtig nach links. Dadurch drückt sich dessen Spitze nach rechts, was zur Folge hat, daß sich die Klappe verbiegt. Wenn Sie nun etwas hebeln, wird sich der linke Teil heben. Sie können die Klappe nun unter leichter Kraftanwendung herausnehmen. Das, was Sie nun sehen, ist ein Teil der Tastaturplatine. Lassen Sie Ihren Blick nach rechts schweifen, können Sie eine Leiste erkennen, die mit vielen Drahtstücken bestückt ist. Da sich dieser Ausdruck etwas unpassend anhört, verwenden wir ab jetzt die richtige Bezeichnung: Pins.

Wenn Sie Ihre Karte in die Hand nehmen, werden Sie - egal von welchem Hersteller sie gebaut wurde - an einer Seite eine Plastikleiste finden, in der sich viele kleine Löcher befinden. Diese Leiste bildet das Gegenstück zur Pinleiste im Amiga. Mit anderen Worten gesagt, muß man diese Leisten aufeinanderstecken. Dadurch ist schon fast klar, wie die Karte eingebaut werden muß. Die Leiste muß nach rechts zeigen. Versuchen Sie noch nicht, die Karte einzusetzen. Denn wenn Sie die Karte einsetzen, müssen die ICs - so nennt man die schwarzen Bauteile - auf der nicht sichtbaren Seite liegen, also unten.

Nehmen wir einmal an, Ihre Karte ist mit einem Schalter zum Abschalten ausgerüstet. Dieser Schalter wird vermutlich an einem mehr oder weniger langen zweiadrigen Kabel hängen. Diesen Schalter wollen Sie bestimmt an einer günstigen Stelle anbringen. Dabei gibt es mal wieder mehrere Möglichkeiten. Die einfache und schlechte Lösung kann man durch ein kleines Loch in der Bodenplatte realisieren. Dieses Loch bringen Sie am Rand der Klappe an. Dazu verwenden Sie am besten eine Feile. Durch diese Einsparung können Sie das Kabel dann herausführen und den Schalter irgendwo ablegen. Beispielsweise hinter dem Computer. Bei der zweiten Möglichkeit verliert man zwar die Garantieansprüche und muß etwas basteln, hat dafür aber eine endgültige Lösung. Besonders dann, wenn man weitere interne Veränderungen vornehmen möchte, ist diese Möglichkeit zu bevorzugen.

Bevor wir uns allerdings damit auseinandersetzen, sollen noch einige kleine Probleme mit den Karten diverser Fremdhersteller besprochen werden. Eins vorweg: Das, was Sie nun lesen werden, hört sich schlimmer an, als es tatsächlich ist. Nehmen wir einen aus dem Leben gegriffenen Fall. Nachdem ich mir meine Speichererweiterung von einem Fremdhersteller gekauft hatte, stellte ich fest, daß diese nicht in den vorgesehenen Platz paßte - sie war ein-

fach ein bißchen zu breit. Mit diesem Problem wird wohl noch der eine oder andere von Ihnen konfrontiert werden. Man kann hier aber leicht Abhilfe schaffen. Notwendig ist eine normale Feile. Hüten Sie sich jedoch, aus Vorsicht eine Nagelfeile zu nehmen, Sie würden viel zu lange feilen müssen, oder eine Raspel zu verwenden, Sie wären sonst womöglich gezwungen Ihre zweite Speichererweiterung zu bestellen, da sich auf der Platine Ihrer ersten durch den rohen Einsatz der Raspel winzige nicht sichtbare Haarrisse gebildet haben könnten, die diese unbrauchbar machen.

An dieser Stelle dazu und auch allgemein ein kleiner Rat: Sollten Sie aus irgend einem Grund eine defekte Platine erhalten haben, die Sie wegen Intoleranz des Händlers oder wegen eines verlegten Kassenzettels nicht umtauschen können, rate ich Ihnen, eine unbestückte Karte zu bestellen und die funktionsfähigen RAM-Chips von Ihrer alten Karte auslöten zu lassen, um sie dann in die neue Karte einzusetzen bzw. einzulöten. So sparen Sie unter Umständen ein- oder zweihundert Mark.

Das Auslöten von Chips ist eine Arbeit, die wirklich nur von erfahrenen Lötprofis durchgeführt werden kann. Fragen Sie aber nach dem Preis, bevor Sie dies von einem Elektroniker machen lassen. Am billigsten kommen Sie wahrscheinlich weg, wenn Sie Lötarbeiten dieser Art in einem kleinen Elektronikladen vornehmen lassen. Suchen Sie nach Möglichkeit kein Geschäft auf, das Computer für den Bürobedarf verkauft! In dieser Branche werden nämlich unerhörte Preise erhoben.

Lassen Sie sich durch das eben gelesene nicht abschrecken - so etwas kommt so gut wie nie vor.

Aber weiter. Sie haben Ihre Karte vorsichtig soweit reduziert, daß sie gerade so in den für sie vorgesehenen Platz paßt. Wenn Sie die Platine einlegen, liegt diese auf zwei Plastikleisten auf, auf denen Sie sie nach rechts schieben können, bis sie an die Kontakte kommt. Mit Gefühl ist sie leicht auf die Pins zu schieben.

Die Karte wäre nun drin. Nun könnten Sie den Deckel wieder zumachen und testen, ob alles funktioniert. Da wir aber keine halben Sachen machen wollen, nehmen Sie bitte den Deckel zur Hand und besehen sich die Seite, die später nach innen kommt. Wenn Sie genau hinsehen, werden Sie zwei kleine Nobben vorfinden. Diese Nobben sollen der Karte zusätzlichen Halt geben. Bei vielen Erweiterungen drücken sie sich aber an einer ungünstigen Stelle gegen

die Platine, so daß die Klappe, nachdem sie eingesetzt wurde, etwas eingedrückt wird. Sollte das der Fall sein oder haben Sie einfach Lust, die Nobben zu entfernen, sollten Sie dies mit einer Kneifzange oder etwas ähnlichem tun. Dann können Sie den Deckel aufsetzen, und Ihre Speichererweiterung zum ersten Mal testen.

Achten Sie darauf, daß Sie die Leitung des eventuell vorhandenen Schalters herausführen. Schließen Sie nun alle wichtigen Stecker wieder an, und starten Sie die Workbench. In der Leiste des Workbench-Screens müßten nun etwa 920000 freie Bytes angezeigt werden. Sollte der Wert im Vergleich zu Ihrer kartenlosen Zeit unverändert sein, kann es sein, daß die Karte nicht richtig auf die Pins geschoben wurde. Haben Sie eine abschaltbare Karte, ist es viel wahrscheinlicher, daß der Schalter in der falschen Position liegt.

Testen können Sie dies, indem Sie ihn umschalten. Nun schalten Sie den Amiga aus, warten eine Weile und schalten ihn dann wieder an. Nun können Sie die Workbench erneut booten. Achtung, Sie müssen den Computer ausschalten, da sonst der neue Speicher nicht anerkannt wird. Merken Sie sich das bitte für alle Ewigkeit! In der Kopfleiste sollte nun jedenfalls die richtige Speichermenge angezeigt werden.

4.1.2 Manchmal nötig: Zerlegung des Amiga

Nachdem Sie nun über eine Million Bytes im Amiga haben, sind Sie für die nächste halbe Stunde wahrscheinlich rundum zufrieden. Das ändert jedoch nichts daran, daß wir jetzt (oder später) noch einmal schwer arbeiten müssen, wenn wir den Amiga noch etwas verbessern oder - um ein moderneres Wort zu verwenden - tunen wollen. Bevor wir anfangen, noch ein Tip: Wir werden Ihnen nun zeigen, wie Sie den 500er komplett zerlegen.

Für uns ist das Zerlegen von Computern eine lästige Routinearbeit. Sie werden das hingegen nicht alle Tage tun. Das sollen Sie auch gar nicht, da dabei leicht mal etwas kapput gehen kann - und Reparaturen sind, wenn man Sie nicht selbst durchführt, meistens sehr teuer. Lesen Sie also das Hardware-Kapitel zu Ende, bevor Sie anfangen, mit uns zu basteln. Wir werden die Demontage des Amiga am Beispiel des Schaltereinbaus für die Speichererweiterung vorführen.

Wenn Sie sich überlegt haben, welche der Veränderungen, die wir Ihnen im Verlauf des Kapitels vorführen, Sie vornehmen wollen, kaufen Sie am besten alle dafür nötigen Teile auf einmal und bauen auch alles auf einmal um. Auf diese Weise müssen Sie den Amiga nur ein einziges Mal auseinandernehmen und wieder zusammenflicken - und zwar so, daß keine Teile übrigbleiben. Das ist übrigens kein Witz sondern ein gut gemeinter Vorschlag. Auch ich habe schon mal das eine oder andere Teile in ein Gerät nicht mehr eingebaut...

Nun kann es aber endlich losgehen. Zuerst müssen alle Stecker abgenommen werden. Dann empfiehlt es sich, die Karte - falls schon vorhanden - auszubauen. Nun drehen wir den Amiga um. Dabei legen wir ihn wieder so, daß die Klappe für die Karte links und die Schnittstellen hinten liegen. Wenn Sie den Amiga leicht anheben, muß die Tastatur vorne liegen. Tut sie das, ist alles in Ordnung.

Nun kommt der schwerste aller Schritte: Das Siegel muß gebrochen werden. Wenn Ihre Garantie schon abgelaufen ist, können Sie das ohne Folgen tun. Andernfalls endet Ihre Garantie in dem Augenblick, in dem Sie mit einem Schraubenzieher das Garantiesiegel durchbrechen. Viele versuchen, das Siegel mit einem Fön abzulösen und nach vollendeter Arbeit wieder anzubringen. Aber das ist jedem selbst überlassen. Zulässig ist es jedoch nicht. Durch die hohen Temperaturen können Sie dem 500er eventuell auch Schaden zufügen. Wasserdampf dürfen Sie selbstverständlich auch nicht verwenden.

Nun lösen Sie die drei vor Ihnen liegenden Schrauben. Legen Sie diese am besten auf ein Blatt Papier, und kreisen Sie sie ein. Nun können Sie den Kreis mit der Position der Schrauben beschriften. Wenn sich dieses Vorgehen auch etwas lächerlich anhört, hat es doch seinen Grund. Wenn Sie nämlich einen Haufen verschiedener Schrauben vor sich haben, kann es leicht sein, daß Sie den Überblick verlieren und die unterschiedlichen Schrauben verwechseln. Dabei können Sie leicht ein Gewinde zerstören. Doch machen wir weiter:

An der hinteren Seite des Amiga sind nochmals drei Schrauben. Aber Vorsicht! Diese Schrauben liegen in einer Reihe über den Ports. Verwechseln Sie sie nicht mit denen, die sich - angeordnet in einem Dreieck - auf der Unterseite des Amiga befinden. Sie halten nämlich das interne Laufwerk.

Wenn Sie nun alle Schrauben gelöst haben, können Sie den Amiga herumdrehen und vorsichtig das Oberteil abheben. Ohne weiteres geht dies nicht. An

beiden Seiten befinden sich nämlich kleine Häkchen. Mit einigen geschickten Griffen ist es aber dennoch zu schaffen.

Jetzt sehen Sie im hinteren Teil des Amiga ein Blech und im vorderen die Tastatur. Von ihr geht ein Kabel aus, das in einer Ausparung des Blechs mündet. Wir können nicht umhin, diesen Stecker herauszuziehen. Ziehen Sie aber bitte nicht an den Drähten. Meistens passiert zwar nichts, aber wie so oft kann es auch hier ein erstes Mal geben. Sehr wichtig ist, daß Sie markieren, wie der Stecker später wieder einzustecken ist!

Nun können Sie die Tastatur einfach abheben. Da wir ja den Schalter für die Speicherkarte einbauen wollen, müssen wir uns jetzt überlegen, wo wir ihn anbringen. Dabei müssen wir bedenken, daß er einserseits gut zugänglich, andererseits aber auch so gelegen sein soll, daß man nicht aus Versehen anstößt. Sehr günstig ist der Bereich rechts neben den Mausports. Hier können sogar mehrere Schalter angebracht werden.

Wir werden es nicht vermeiden können, dazu den Amiga komplett auseinanderzunehmen. Wenn Sie sich dabei genau an unsere Anleitung halten, kann eigentlich gar nichts schief gehen. Zunächst muß das große Blech weg. Dazu müssen wir zuerst ein kleines Blechstück über dem Expansionport entfernen. Es ist mit zwei Kreuzschlitzschrauben befestigt. Vermeiden Sie, wenn irgend möglich, daß dieses Blech auf die Leiterbahnen des Erweiterungsports fällt. Danach kommt das große Blech dran. Es ist vorn mit zwei Schrauben befestigt. Zusätzlich wird es durch einige umgepfalzte Blechteile gehalten. Nachdem Sie die Schrauben gelöst haben, müssen Sie nachsehen, an welchen Stellen die umgebogenen Blechteilchen aufzurichten sind. Mit umgebogenen Blechecken ist das so eine Sache. Vermutlich sollten alle diese Laschen dem Abschirmblech zusätzlichen Halt geben. Es kann aber sein, daß nicht alle von ihnen richtig umgebogen wurden. Um die Abschirmung zu entfernen, müssen jedenfalls alle sichtbaren Blechanhängsel auch so aufgebogen werden, daß sie senkrecht stehen.

Jetzt kann die Abschirmung endlich abgehoben werden. Im Bereich des Laufwerks ist etwas Gefühl angesagt.

Wenn Sie einige Blechlaschen aufrichten mußten, biegen Sie diese bitte wieder zurück, und zwar so weit, daß sie waagrecht liegen. So reduzieren Sie die Verletzungsgefahr. Später brauchen Sie die Abschirmung nicht wieder durch

erneutes Verbiegen der Laschen zu befestigen. Sie hält auch so. Auch können die Laschen abbrechen, wenn sie zu oft umgebogen werden.

Jetzt können Sie endlich die Platine sehen. Sie scheint extrem einfach gebaut zu sein. Der Grund dafür ist aber nicht, daß der Amiga wenig Fähigkeiten hätte. Vielmehr sieht die Platine so leer aus, um den Amiga billig produzieren zu können. Die vielen kleinen Teile, die normalerweise auf einer Computerplatine zu finden sind, wurden in wenigen großen zusammengefaßt.

Das fette Bauteil rechts neben dem Expansionport ist der 68000er - seines Zeichens Prozessor und Chef des Systems. Zu seiner Rechten ist ein etwas kleinerer Chip zu sehen, der auch sehr wichtig ist. Die Rede ist hier vom Kickstart-ROM. In diesem Baustein befinden sich alle Routinen des Kickstart 1.2. Oberhalb von ROM und 68000er liegt Denise. Denise ist der Name des Chips, der für die Bildschirmdarstellung und einige andere Dinge zuständig ist. Etwas weiter rechts finden Sie noch Paula und Gary - zwei weitere speziell für den Amiga entwickelte Chips. Paula ist für die Kommunikation mit den unterschiedlichsten Geräten zuständig. Gary hingegen ist etwas ganz besonderes. Er beinhaltet viele kleine Funktionen, die sonst die Platine füllen würden. Ziemlich nah an den Ports sind noch zwei weitere ICs zu finden. Sie tragen sie Bezeichnungen ODD CIA und EVEN CIA. Gebraucht werden Sie, um einige Schnittstellen zu bedienen. Und wo ist der Speicher? Im vorderen Teil der Platine sehen Sie ein Feld mit kleinen ICs, die in zwei Reihen angeordnet sind. Diese ICs enthalten die 512 KByte RAM. Nun fehlt nur noch einer. Der Blitter. Er ist zusammen mit dem Copper und einigen anderen Dingen in einem Chip namens FAT AGNUS untergebracht. Der Copper ist übrigens auch ein Prozessor. Er ist jedoch nur für Trickdarstellung vorgesehen. Auch sorgt er dafür, daß Sie verschiedene Screens mit unterschiedlichen Auflösungen darstellen können.

Aus dem Laufwerk führen zwei Kabel, von denen eins sehr breit ist. Solche breiten Kabel nennt man in der Fachsprache meist Flachbandkabel. Unsere Aufgabe ist es nun, die Kabel von der Platine zu lösen. Dabei müssen wir uns merken, wie die Stecker später wieder anzubringen sind. Leichtes Spiel haben wir bei dem kleinen Stecker. Er kann einfach herausgezogen werden. Schwierig wird es aber beim Flachbandkabel. Versuchen Sie in keinem Fall am Kabel direkt zu ziehen. Wenn es Ihnen nicht gelingt, den Stecker mit den Fingern von der Platine zu lösen, können Sie sich eines kleinen Tricks bedienen: Wenn Sie sich die Verbindung des breiten Steckers mit der Platine genau ansehen, können Sie bemerken, daß der Stecker - eigentlich ist es übrigens eine

sogenannte Kupplung - etwas breiter ist als sein Gegenstück auf der Platine. Weiterhin liegt er ziemlich nah am hinteren Gehäuserand. Ganz vorsichtig können Sie nun abwechselnd links und rechts einen Schraubenzieher unter den überstehenden Bereich schieben. Er darf die Platine auf keinen Fall berühren. Wenn Sie ihn nun auf dem Rand des Gehäuses auflegen, können Sie den Stecker etwas hochhebeln. Auf jeder Seite sollten Sie dies immer nur ein bißchen tun. So wird gewährleistet, daß sich die Pins nicht verbiegen. Denken Sie aber immer wieder daran, den Schraubenzieher von der empfindlichen Oberfläche der Platine fernzuhalten.

Nun kommt das Laufwerk an die Reihe. Es wird von vier Schrauben gehalten. Drei der Schrauben befinden sich unter dem Amiga und halten es von unten. Eine weitere ist seitlich angebracht. Wenn Sie das offene Gerät von vorne betrachten, liegt diese Schraube rechts am Floppy. Diese Schraube ist zuerst zu lockern. Ganz rausdrehen muß man sie nicht. Wenn die verbleibenden drei Befestigungen gelöst sind, kann man das Laufwerk herausheben. Legen Sie es aber so ab, daß die Seite, die normalerweise oben ist, nach unten zu liegen kommt. Auf der eigentlichen Unterseite liegt nämlich die Mechanik frei.

Vorausgesetzt, Sie haben keine Karte installiert, können Sie die Platine nun aus dem Gehäuse heben. Fassen Sie dazu am besten die Randstücke des Schutzblechs an.

Nun haben Sie endlich alles auseinandergebaut. An der vorgesehenen Stelle kann nun das Loch für den Schalter gebohrt werden. Wenn Sie noch andere Schalter anbringen wollen, sollten Sie dafür schon die Löcher anbringen. Auch wenn Sie die zugehörigen Erweiterungen nocht nicht haben. So ersparen Sie sich unter Umständen später viele Arbeitsschritte.

Die Schalter der Speicherkarten sind meist so gebaut, daß sie in einem runden Loch gut befestigt werden können. Um das Loch sauber anzubringen, empfehle ich Ihnen einen elektrischen Stahlbohrer. Fangen Sie mit einer kleinen Größe an, und vergrößern Sie den Bohreinsatz nach und nach. So erzielen Sie, wenn Sie von außen bohren, ein sauberes Loch. Natürlich dürfen Sie nicht vergessen, nach jeder Bohrung zu kontrollieren, ob der Schalter schon paßt. Ein Millimeter zuviel kann Ihnen viel Ärger bereiten. Deshalb sollten Sie die Bohreinsatzstärke auch immer nur um einen halben Millimeter erhöhen. Paßt der Schalter dann, können Sie die Fransen, die eventuell innen aufgetreten sind entfernen und das Gehäuse von herumfliegenden Spänen säubern. Dann kann der Einbau beginnen.

Zuerst ist die Platine einzusetzen. Dann können Sie die Karte von unten einlegen und vorsichtig einschieben. Den Schalter befestigen Sie jetzt in Ihrem Bohrloch. Kleben Sie das Kabel, das zu ihm führt auf alle Fälle im Bereich unter dem Laufwerk am Boden des Gehäuses fest, da es sonst leicht in die Mechanik des Laufwerks geraten kann. Nun kann das Laufwerk wieder an seinen Platz gesetzt werden. Achten Sie darauf, daß seine Füßchen nicht auf die grünen Teile der Platine kommen. Sie müssen auf die beiden silbernen Rechtecke gesetzt werden. Im nächsten Schritt wird das Laufwerk befestigt. Die beiden Kabel dürfen auch nicht vergessen werden. Das Flachbandkabel läßt sich übrigens leicht aufstecken. Anstrengungen, wie sie beim Ausbau nötig waren, müssen jedenfalls nicht aufgebracht werden. Nachdem die Abschirmung aufgesetzt und angeschraubt ist, muß das kleine Blech über dem Expansionsport wieder befestigt werden. Das erfordert etwas Fingerspitzengefühl, ist aber ohne weiteres zu bewerkstelligen.

Nächster Schritt ist das Einsetzen der Tastatur. Überprüfen Sie, ob die Folie richtig sitzt, die die Unterseite der Tastaturplatine vom Blechteil isoliert. Sitzt die Tastatur richtig, müssen Sie nur noch deren Stecker wieder anstecken, den Gehäusedeckel aufsetzen und schließlich die sechs Schrauben, mit denen er zu befestigen ist, wieder anbringen.

4.1.3 Gigantisch - Aufrüstung auf 2.2 MByte

Wer keine halben Sachen machen will, kann seinen 500er auch intern auf 2.2 MByte aufrüsten. Das entspricht dem 4.5fachen der normalen Speicherkapazität. Allerdings ist dieser Spaß nicht gerade billig. Wer den großen Speicher braucht, kommt aber preislich relativ gut weg. Die Erweiterung wird von der Firma Gigatron hergestellt und verkauft.

Der Preis der Karte beträgt etwa 1200 DM. Sie ist mit einer Echtzeituhr ausgestattet und kann auch abgeschaltet werden.

Zum Einbau der Karte muß man glücklicherweise nicht löten. Es ist aber nicht zu vermeiden, den Amiga auseinanderzunehmen. Dabei müssen Sie allerdings nicht ganz so weit gehen wie in Kapitel 4.1.2. Es reicht aus, wenn Sie an die Chips der Platine herankommen. Die eigentliche Karte wird wie auch die normale Erweiterung auf ein MByte in den Schacht auf der Unterseite des Amiga gesteckt. Zusätzlich muß man noch einen Chip auf der Platine mit ei-

nem sogenannten Zwischensockel versehen. Es handelt sich um Gary. Gary ist unter anderem für die Speicherverwaltung zuständig. Ein Zwischensockel ist ein kleines Gerät, das in den Steckplatz eines Chips gesetzt wird. Der Chip, der diesen Sockel vorher belegte, wird dann auf den Zwischensockel aufgesetzt. Man wendet dieses Verfahren an, um bestimmte Pins des Chips mit Signalen zu versorgen oder Signale von dort abzunehmen.

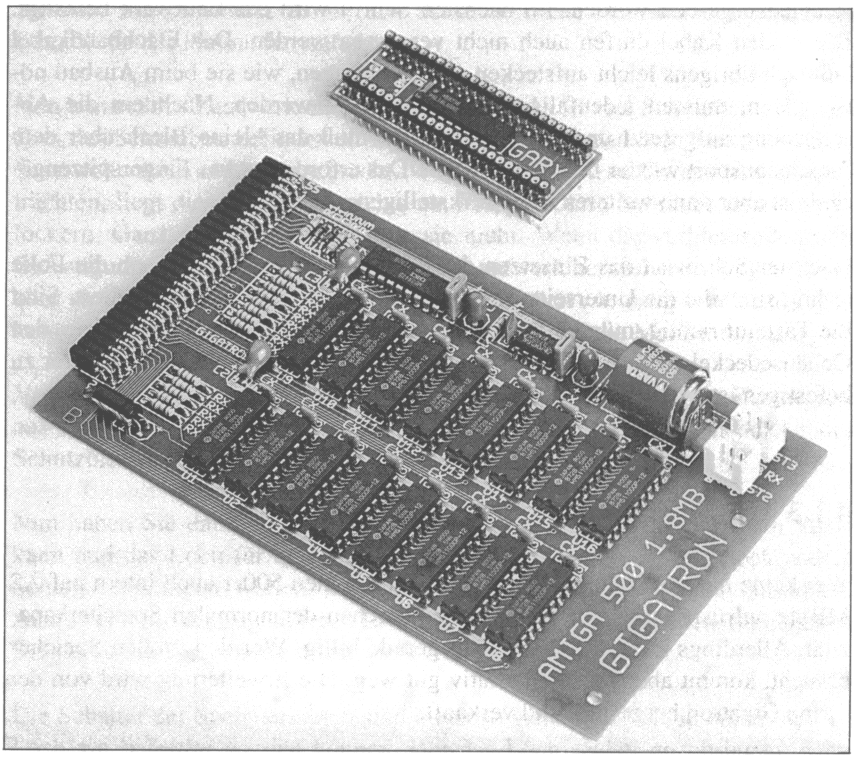


Abb. 4.1: Die Gigatron-Erweiterungskarte

Von diesem Sockel geht ein Kabel zu einem kleinen Stecker, der in die eigentliche Karte einzustecken ist. Der Schalter zum Abschalten der Karte ist nicht, wie es bei anderen Herstellern üblich ist, über ein Kabel fest mit der Karte verlötet, sondern (natürlich auch über ein Kabel) mit einem Stecker verbunden, den man auf ein entsprechendes Gegenstück auf der Speicherplatine stecken kann.

Dieser auf den ersten Blick unscheinbare Vorteil ist unter Umständen jedoch sehr angenehem: man kann die Karte (zu welchem Zweck auch immer) ausbauen, ohne den Schalter zu demontieren und den Gary-Sockel zu entfernen, weiterhin kann man, eventuell auch ohne Löten zu müssen, das Kabel zum Schalter mit einem Zwischenstück verlängern, um ihn auch an weiter entfernten Stellen zu installieren. Allgemein kann man sagen, daß die Karte sehr sauber gebaut und konstruiert wurde.

Bei keiner anderen Speichererweiterung fanden wir z.B. eine Pinleiste mit den (teureren) Präzisionsbuchsen.

Bastler können die Karte auch in Form eines Bausatzes kaufen, der, da er ohne RAM-Bausteine geliefert wird, erheblich billiger ist. Die Karte benötigt 16 Ein-Mega-Bit-Chips vom Typ 511000 mit einer Zugriffszeit von 120 nS.

Den (werdenden) Profis unter Ihnen sei gesagt, daß der neue Speicher der Karte ab $C00000 liegt. Da dies auch die Adresse normaler Erweiterungen ist, ergeben sich in diesem Punkt keine Kompatibilitätsprobleme. Mögliche Defekte an den RAM-Bausteinen lassen sich übrigens mit dem gratis mitgelieferten Programm RAM-Test feststellen - ein Service, den wir auch bei keiner anderen Firma gefunden haben. Bei einem Preis von über 1000,- Mark sollte man so etwas aber schon erwarten.

Um das RAM-Test-Programm zu laden, muß der Speicher mit unserem neuen NoFastMem aus Kapitel 2.9 abgeschaltet werden. Möchten Sie unser Programm nicht abtippen, können Sie sich an die Methode halten, die in der Anleitung vorgeschlagen wird:

1) Rechner ausschalten
2) Karte mittels Schalter ausschalten
3) Amiga einschalten und auf Hand warten!!!
4) Karte einschalten
5) RAM-Test-Diskette einlegen

Sie fragen sich, warum dieser Aufwand betrieben werden muß? Auf diese Frage wollen wir Ihnen natürlich keine Antwort schuldig bleiben. Also: Das RAM-Test-Programm beschreibt die einzelnen Speicherstellen, wartet eine einstellbare Zeit und vergleicht dann den Soll- mit dem Ist-Inhalt. Würden also Programme und Daten im Fast-RAM liegen, würden diese gelöscht, was

zur Folge hätte, daß sich der Amiga mit einem Guru verabschiedet. Der Speicher muß also abgeschaltet werden (2). Wenn er aber abgeschaltet ist, kann man ihn nicht testen. Man muß also warten, bis das Betriebssystem getestet hat, welche Speicherbereiche es für sich beanspruchen darf (3). Wenn die Hand mit der Diskette angezeigt wird, ist dies der Fall (3). Nun kann man die Karte wieder einschalten (4). Sie wird dann vom System nicht beschrieben, kann aber von der RAM-Test-Routine erreicht werden!

Daß sich die Anschaffung der teuren Speicherkarte lohnt verdeutlichen zwei Zahlen: zunächst wird in der Kopfzeile der Workbench eine Speichermenge von mehr als 2000000 Bytes angezeigt. Weiterhin kann man mit einer Textverarbeitung (z.B. BECKERtext) viermal mehr Seiten im Speicher bearbeiten als mit einem MByte. Von dem Unterschied zu Rechnern mit 512 KByte wollen wir erst gar nicht reden.

Gigatron bietet auch eine Karte an, die den Speicher auf satte 4.3 MByte erweitert. Diese Karte hat allerdings einen kleinen Nachteil: Sie ist nicht vollständig autokonfigurierend, d.h. ein Teil des Speichers wird vom Amiga nicht erkannt. Um die volle Kapazität der Karte auszunutzen, muß man den Speicher also mit einem kleinen Programm einbinden.

In Anbetracht der zur Zeit sehr hohen Preise für RAM-Chips ist es ohnehin ratsam, mit dem Kauf einer Speicherkarte dieser Größenordnung noch etwas zu warten. Auch kommt man eventuell billiger weg, wenn man man den Amiga 500 verkauft und durch einen 2000er ersetzt, den man dann mit relativ billigen Steckkarten auf 4 MByte erweitern kann. Auch der Bausatz aus Kapitel 4.7 dürfte in diesem Zusammenhang interessant sein. Die Speicherkarte, die den Amiga 500 auf 2.2 MByte aufrüstet, kann man jedoch jedem ernsthaften Anwender empfehlen.

4.2 'ne Disk dazu

Ständig verlangt er von uns, irgend eine Diskette einzulegen. Ist denn sein Speicher nicht groß genug, um alle Daten, die benötigt werden, dort abzulegen? Dieser Gedanke wird wohl dem einen oder anderen von Ihnen schon einmal durch den Kopf gegangen sein. Und das mit Recht. Der Amiga arbeitet nämlich sehr intensiv mit seinen Speichermedien. Und da es nicht gerade angenehm ist, ständig Disketten zu wechseln, besteht die Möglichkeit, drei zu-

sätzliche Laufwerke an den Rechner anzuschließen, die dann (dank der modernen System-Konzeption) auch von fast allen Programmen angesprochen oder wenigsten benutzt werden können.

Wer der Ansicht ist, daß ein zusätzliches Diskettenlaufwerk einfach angeschlossen wird und dann seinen Dienst verrichtet, ist einerseits richtig, andererseits aber auch falsch informiert. Richtig ist, daß der Anschluß zusätzlicher Drives, wie man Laufwerke auch zu bezeichnen pflegt, kinderleicht vorzunehmen ist. Falsch ist jedoch, daß man dann automatisch mit dem oder den neuen Laufwerken arbeiten oder es/sie gar voll ausnutzen kann.

Es gibt nämlich viele Dinge, die man bei der Anwendung und sogar schon beim Kauf beachten sollte. Beginnen wir daher mit den Überlegungen, welches Laufwerk wir brauchen. Dabei unterscheiden wir verschiedene Kriterien.

Preis-/Leistungsverhältnis

Dieses Entscheidungskriterium ist nicht gerade einfach zu handhaben. Als Käufer läßt man sich nur zu gerne von extrem billigen Angeboten locken. Andererseits möchte man aber auch gute Qualität und einige Extras bekommen. Bevor man daher die Preise der einzelnen Hersteller vergleicht, sollte man erst einmal genau überlegen, was das neue Laufwerk leisten soll. Erst dann sollte man in diversen Anzeigen nach dem billigsten Laufwerk suchen, das diese Leistungsmerkmale aufweist. Wenn Sie auf "schwammige" Formulierungen in den Aussagen der Hersteller treffen sollten, ist es ratsam, sich vor dem Kauf schriftlich bestätigen zu lassen, was man wissen möchte.

Anschlußkabel und durchgeführter Bus

Wenden wir uns nun dem ersten konkreten Punkt der Kaufentscheidung zu: dem Anschluß-Kabel. Wichtig ist hier vor allem, daß das Kabel lang genug ist, um das Laufwerk an der Stelle postieren zu können, an der man es haben möchte. Weiterhin sollte man darauf achten, daß das Gerät einen "durchgeschleiften bzw. durchgeführten Bus" hat. Dieser Ausdruck im Computer-Chinesisch besagt nichts anderes, als daß an dem Laufwerk eine Buchse vorhanden ist, an die ein weiteres Drive angeschlossen werden kann.

Ausschalter

Auch über einen Schalter, mit dem das Laufwerk ausgeschaltet werden kann, sollte jedes Gerät verfügen. AmigaDOS belegt nämlich für jedes angeschlossene Laufwerk einen Speicherblock im Chip-RAM. Da einige Programme diesen Speicher voll und ganz benötigen, kann es sein, daß man zusätzliche Laufwerke abschalten muß, wenn man das Programm verwenden will. Einige dieser speicherplatzintensiven Programme lassen es jedoch zu, daß das Laufwerk angeschaltet beibt, wenn zusätzliches Fast-RAM bereitsteht.

Zustands-LED

Eigentlich selbstverständlich ist eine Leuchtdiode, die anzeigt, ob das Laufwerk gerade auf die Diskette zugreift. Bei einem Gerät, das keine LED besitzt, die dies signalisiert, besteht die Gefahr, daß man eine Diskette aus dem Laufwerk nimmt, bevor alle Daten korrekt und vollständig gespeichert wurden.

Einschaltbarer Schreibschutz

Nicht unbedingt notwendig, aber gelegentlich sehr nützlich ist ein sogenannter Write-Protect-Schalter. Er macht es möglich, dem Amiga eine Diskette, die eigentlich nicht schreibgeschützt ist, als schreibgeschützt zu präsentieren. Man kann auf diese Weise verhindern, daß Daten auf Disketten im externen Laufwerk aus Versehen überschrieben werden. Doch nicht nur das. Einige Kopierprogramme verlangen, daß die Quelldiskette schreibgeschützt ist, um zu verhindern, daß beim Wechseln der Disketten aus Versehen Daten auf der Quell-Diskette zerstört werden, weil sie versehentlich mit der Ziel-Diskette vertauscht wurde. Schaltet man den Write-Protect-Schalter ein, erübrigt es sich, für jede Diskette, die man Kopieren will, den Schreibschutz einzustellen, wenn man die Quell-Diskette in das externe Laufwerk legt.

Lautstärke

Über diesen Punkt schweigen sich die Anbieter von externen Laufwerken meist aus. Dennoch ist er oft sehr wichtig, denn auch dann, wenn der Amiga

nicht auf ein Laufwerk zugreift, um Daten zu laden, gibt dieses doch alle paar Sekunden einen "Knacks" von sich, weil das Betriebssystem nicht direkt feststellen kann, ob eine neue Diskette eingelegt ist, und es daher einen Scheinzugriff durchführen muß. Wenn man nicht nur ein einziges sondern gleich mehrere Laufwerke angeschlossen hat, knackt es auch etwas häufiger. Bei sehr lauten Geräten hilft dann nur noch die Notlösung: alle Laufwerke mit Disketten "vollstopfen", nur um die Geräuschentwicklung einzudämmen - das kann nicht der Sinn der Sache sein. Das uns vorliegende 3½-Zoll-Golem-Drive war jedoch vorbildlich leise.

Einstellbare Gerätenummer oder Automatik

Stellen Sie vor dem Kauf sicher, daß Sie das Laufwerk auch als Drittgerät benutzen können. D.h. es muß in der Lage sein, auch dann noch einwandfrei zu arbeiten, wenn es nicht direkt, sondern über ein oder zwei andere Laufwerke mit dem Rechner verbunden ist. Dies ist heute zwar Standard, doch besonders bei kleinen Firmen kann eine Anfrage nichts schaden. Auch wenn für Sie feststeht, daß Sie nie ein drittes Laufwerk kaufen werden, kann es immerhin sein, daß Sie Ihr Gerät einmal mit zu einem Bekannten mitnehmen, der ebenfalls ein externes Drive angeschlossen hat, an das Sie Ihr Laufwerk noch anhängen wollen, um beispielsweise mit mehreren Ziel-Disketten gleichzeitig kopieren zu können.

Track-Anzeige

An dieser Stelle wollen wir uns nicht damit beschäftigen, was eine Track-Anzeige ist und wer sie benötigt. Wir wollen lediglich darauf hinweisen, daß es unter Umständen sinnvoll ist, die Track-Anzeige gleich im Laufwerk integriert zu haben.

Einzeln oder im Verbund

Will man gleich zwei Laufwerke ergänzen, stellt sich die Frage, ob man die Geräte getrennt oder in einem Gehäuse kauft. Wir wollen an dieser Stelle noch nicht näher auf dieses Thema eingehen, da wir erst noch einige Grundla-

gen klären müssen, um hier sachkundig entscheiden zu können. Wir weisen an dieser Stelle nur deshalb schon auf das Problem hin, um zu vermeiden, daß Sie, bevor Sie das ganze Kapitel zu Ende gelesen haben, eine Bestellung aufgeben, die Sie noch bereuen könnten.

Anschluß des Laufwerks

Der Anschluß des Laufwerks ist sehr leicht vorzunehmen: An der Rückseite des Amiga gibt es eine Buchse, die für den Anschluß von Diskettenlaufwerken vorgesehen ist. Sie liegt zwischen den beiden Tonausgängen und der seriellen Schnittstelle. An ihr wird der Stecker des ersten zusätzlichen Laufwerks angeschlossen. Hat dieses Laufwerk einen durchgeführten Bus, findet man an seiner Rückseite eine Buchse, an der man dann weitere Laufwerke anschließen kann. Dabei sollte man allerdings beachten, daß die Leitungen des Floppy-Steckers nicht überlastet werden dürfen. Schließt man mehrere Laufwerke an, kann es sein, daß man solche Geräte kaufen muß, die mit einem eigenem Netzteil geliefert werden. Von der Möglichkeit, den Laufwerksstecker festzuschrauben, sollten Sie, wenn möglich, Gebrauch machen. Sie verhindern auf diese Weise, daß durch einen Wackelkontakt wichtige Daten verloren gehen.

Zur Zeit im Trend: Track-Anzeigen

Mittlerweile sind Track-Anzeigen zu einer Mode-Erscheinung geworden. Alle Welt redet davon, doch was versteht man eigentlich darunter? Eine Diskette kann bekanntlich in mehrere Spuren (= Tracks) unterteilt werden. Beim Amiga sind es 80 an der Zahl, die von 0-79 durchnumeriert sind. Eine Track-Anzeige stellt nun auf einer zweistelligen Digitalanzeige die Nummer des Tracks (der Spur) dar, über der sich der Schreib-/Lesekopf gerade befindet.

In den meisten Fällen dient diese Anzeige nur dazu, dem Anwender die Wartezeiten zu verkürzen, in denen er darauf wartet, daß ein Programm endlich geladen ist. Kennt man sich aber etwas mit dem Aufbau einer Diskette aus, kann man aus bestimmten Bewegungen des Schreib-/Lesekopfes bestimmte Schlüsse ziehen. Nicht nur für Cracker, also Personen, die den Kopierschutz von Software entfernen, dürfte die Track-Anzeige interessant sein. Auch normalen Anwendern, die gezwungen sind, eine Diskette zu restaurieren, hilft die

Track-Anzeige oft weiter. Bei externen Laufwerken gibt es zwei Möglichkeiten, Track-Anzeigen zu verwenden. Entweder kann man schon beim Kauf des Laufwerks darauf achten, daß eine Track-Anzeige mit eingebaut ist (das ist die billigere und elegantere Methode) oder man kann eine externe Trackdisk-Anzeige anschließen, die auch für das interne Laufwerk benutzt werden kann. Solche externen Track-Anzeigen werden von verschiedenen Firmen (z.B. Alcomp) angeboten.

Neben der Spur zeigt das Gerät von Alcomp mit zwei LEDs noch an, welche Seite der Diskette gerade bearbeitet wird. Der Amiga beschreibt seine Disketten ja bekanntlich beidseitig. Es hat deshalb auch keinen Sinn, zu versuchen, eine Diskette herumzudrehen, um auch die andere Seite verwenden zu können, wie dies beim C64 und anderen Rechnern üblich war.

Die Golem-Drives, die uns die Firma Kupke freundlicherweise zur Verfügung gestellt hatte und die wir während der Arbeit an diesem Buch verwendet haben, verfügten auch über eine Track-Anzeige. Auch bei ihnen wurde angezeigt, welcher Schreib-/Lesekopf gerade verwendet wird. Dazu sind allerdings keine besonderen LEDs vorhanden. Man bediente sich hier eines kleinen Tricks, indem man die Dezimalpunkte auf den beiden 7-Segment-Leuchtanzeigen verwendete. Der eben verwendete Fachbegriff begründet sich übrigens in der Tatsache, daß Digitalzahlen aus maximal sieben Strichen zusammengesetzt werden.

4.2.1 Die Alternative: 5¼-Zoll-Laufwerke

Oft werden sie schon zum alten Eisen gezählt: die 5¼-Zoll-Disketten und die dazugehörigen Laufwerke. Die Disketten sind biegsam und der Bereich, in dem der Lesekopf des Laufwerks die Daten liest, ist nicht wie bei den 3½-Zoll-Disketten mit einem Metallschieber geschützt. Dafür werden sie in Hüllen aufbewahrt, um sie vor Staub zu schützen. Angeblich können sie auch nicht so viele Daten aufnehmen wie ihre kleineren Verwandten.

Über die Vor- und Nachteile der verschiedenen Diskettengrößen kann man sich sicher streiten - in einem Punkt ist jedoch jeder Zweifel ausgeschlossen: im Preis! 5¼-Zoll-Disketten kosten oft nur die Hälfte oder ein Drittel dessen, was man für die Kleinen hätte blechen müssen. Auch sind sie im allgemeinen recht zuverlässig.

Wenn man den Kostenfaktor mit einbezieht, stellt sich die berechtigte Frage, ob es nicht möglich ist, ein oder mehrere große Floppies an den Amiga 500 anzuschließen.

Diese Gedanken hatten wohl auch einige Hardware-Hersteller, die bewährte Laufwerke, meistens vom japanischen NEC-Konzern, in ein stabiles Gehäuse packten und auf den Amiga-Markt warfen.

Ursprünglich sollten die 5¼-Zoll-Laufwerke am Amiga nur 40 Spuren schreiben können. Da dies technisch gesehen jedoch weit unter dem lag, was machbar war, kamen schon bald Geräte auf den Markt, die 80 Tracks bearbeiten konnten. Damit waren sie in der Lage, genauso viele Daten zu speichern wie ihre kleinen Kollegen.

Jetzt wird es aber erst richtig interessant. Früher erkannten die großen Laufwerke keinen Diskettenwechsel. Heute beherrschen sie jedoch auch diese Technik - sie sind also den 3½-Zoll-Geräten zu gleichwertigen Konkurrenten geworden. Da der Amiga nicht mehr unterscheiden kann, ob er auf eine große oder eine kleine Diskette zugreift, ist es sogar möglich, kopiergeschützte Software auf das neue Diskettenformat umzukopieren. (Ausnahmen könnten hier in der Zukunft auftreten.) Es besteht nun die Möglichkeit, die Software-Sammlung auf billigere Disketten zu übertragen, so daß es sein könnte, daß sich das 5¼-Zoll-Laufwerk von selbst finanziert.

Vor dem Kauf sollten Sie sich allerdings beim Hersteller erkundigen, ob das entsprechende Gerät auch in der Lage ist, einen Diskettenwechsel festzustellen. Beim unserem Gerät war dies der Fall. Es könnte jedoch sein, daß andere Hersteller diese Technik noch nicht realisiert haben.

Als Ergänzung: Der Bootselector

Hat man seine Software auf 5¼-Zoll-Disketten umkopiert, steht man vor dem Problem, daß es machmal nötig ist, eine Programm-Diskette zu booten, d.h. ein Programm mittels Reset zu starten.

Da der Amiga die System-Diskette nach einem Reset in DF0: erwartet, muß man ihm klar machen, daß nun das erste externe Laufwerk, in diesem Fall das 5¼-Zoll-Drive DF0: heißen soll. Auch hierzu bietet die Firma Alcomp wieder

ein kleines Hilfsmittel an, das diesmal allerdings in den Amiga selbst eingebaut werden muß. Ist dies geschehen, kann man mit einem Schalter festlegen, welches Laufwerk DF0: sein soll.

Um den Bootselector (so wird das Gerät genannt) zu installieren, müssen wir unseren 500er wieder, wie in Kapitel 4.1.2 beschrieben, öffnen. Dann hebeln wir vorsichtig den Chip heraus, der unter dem Flachbandkabel der internen Floppy verborgen liegt. Auf der Platine ist er als U8 und/oder EVEN CIA gekennzeichnet. Auf dem Chip selbst müßte die Typenbezeichnung 8520 stehen. In den Sockel auf der Platine setzen Sie nun den Bootselector ein. Er besteht eigentlich nur aus ein paar aufeinandergelöteten Sockeln, an denen einige Verbindungen mit Hilfe eines Schalters, der an der Gehäuserückwand zu installieren ist, hergestellt werden können. Auf den Selector setzen Sie wieder den IC ein, der vorher direkt im Sockel auf der Platine steckte. Alle drei Kerben (die des Sockels, des Selectors und des Chips) müssen in die gleiche Richtung zeigen. Dann können Sie das Floppy-Kabel wieder einstecken. Unter Umständen kann dies etwas schwierig sein, da es nun über den Sockelturm gebogen werden muß.

Mögliche Fehlerquelle

Der Bootselector bietet eine kleine Tücke, an die man sich jedoch gewöhnen kann. Genaugenommen tauscht er nämlich zwei Laufwerke aus. Folgender Fall: DF0: und DF1: sind vorhanden. Wenn Sie nun den Selector umschalten, müssen Sie, wenn Sie das interne Laufwerke ansprechen wollen, DF1: benutzen. Es ist also keinesfalls so, daß in Verbindung mit dem Bootselector ein Laufwerk ausfällt.

Was beim Kauf eines Bootselectors zu beachten ist

Alle mir bekannten Bootselectoren sind nicht in der Lage, mehr als eine Bootalternative zuzulassen. In Klartext heißt das, daß ein Bootselector, der DF1: und DF0: austauscht, nicht in der Lage ist, auch DF0: und DF2: auszutauschen. Man kann (und sollte) daher bei Alcomp bei der Bestellung angeben, auf welches Laufwerk man umschalten will. Möglich sind die folgenden Kombinationen:

```
DF0: DF1:
DF0: DF2:
DF0: DF3:
```

Wer braucht den Bootselector?

Prinzipiell gibt es zwei sinnvolle Einsatzmöglichkeiten für den Bootselector. Zum einen ermöglicht er es, von 5¼-Zoll-Disketten zu booten, zum anderen kann es als billige Alternative zu einer externen Track-Anzeige verwendet werden. Und das geht folgendermaßen: Angenommen, Sie wollen die Bewegungen des Lesekopfes des Laufwerks DF0: untersuchen. Wenn Sie keine Track-Anzeige besitzen, mit der Sie dies für das interne Laufwerk anzeigen könnten, und zufällig ein externes Laufwerk mit Track-Anzeige besitzen, können Sie den Bootselector dazu einsetzen, die Funktion von DF0: auf das externe Laufwerk umzulegen.

Im allgemeinen wird man den Bootselector jedoch nur dazu benutzen, um von einem externen 5¼-Zoll-Laufwerk zu booten. Meiner Meinnug nach hat es nicht viel Sinn, einen Bootselector zu kaufen, wenn man kein 5¼-Zoll-Drive besitzt.

4.2.2 Anpassen der Software

Nach der Anschaffung eines zweiten Diskettenlaufwerks ist es sinnvoll, das eigene Arbeitsverhalten zu ändern. Anfänglich neigt man noch immer zum Diskettenwechseln. Doch auch die System-Software und Anwendungsprogramme sollten angepaßt werden. Und zwar so, daß es nicht mehr nötig ist, immer dann neu zu booten, wenn man ein anderes Programm benutzen will. Bisher war dies noch oft nötig.

Ein Beispiel macht es deutlich: TEXTOMAT, eine Textverarbeitung, die zur Zeit mit allen Amiga 500 ausgeliefert wird, lädt aus FONTS: einige Zeichensätze nach, die auf der normalen Workbench nicht vorhanden sind. Hat man also von dieser Workbench gebootet, kann man nicht einfach die TEXTOMAT-Diskette einlegen und das Programm-Icon anklicken. Das Programm würde nämlich mit der Fehlermeldung abbrechen, daß die Zeichensätze nicht gefunden werden können. Sicher, man könnte die Fonts der Textverarbeitung

auf die Workbench-Diskette kopieren. Doch dann müßte man häufig Disketten wechseln. Anders sieht es aus, wenn mit zwei Laufwerken gearbeitet wird. Liegt die Workbench im einen und die Programm-Diskette im anderen Laufwerk, ist es nicht weiter schlimm, wenn ein Programm abwechselnd auf die eine und die andere Diskette zugreift. Das häufige Diskettenwechseln entfällt.

Bei Programmen, die Sondereinstellungen benötigen oder nur über das CLI aufgerufen werden können, kann man sich mit einigen kleinen Batch-Dateien behelfen, die man im S:-Verzeichnis der Boot-Diskette ablegt. Dort sucht sie nämlich der Execute-Befehl, wenn er im aktuellen Verzeichnis nicht fündig wird. Versieht man dieses Verzeichnis mit einem Icon, kann man, wenn man die Files mit IconX-fähigen Icons versieht (s. Kapitel 2; CLI), auch einen Aufruf über die Workbench realisieren.

Beurteilung der von uns verwendeten Geräte

Golem-Drives

Unsere Golem-Drives haben während der Entwicklungszeit des Buches treu ihren Dienst verrichtet. Für die Verwendung als PC-Laufwerke scheinen sie jedoch nicht geeignet zu sein. Die Track-Anzeige arbeitete auch zuverlässig.

Track-Anzeige von Alcomp

Arbeitet, wenn man außer dem internen kein weiteres Laufwerk angeschlossen hat, absolut zuverlässig. Hat man jedoch mehrere externe Geräte angeschlossen, wird manchmal ein falscher Wert dargestellt.

Bootselector von Alcomp

Das Gerät erfüllte seinen Zweck. Der Ausbau aus dem Computer machte aufgrund des verwendeten Sockeltyps (Billigausführung) jedoch Schwierigkeiten. Ansonsten waren wir zufrieden.

Zusammen oder getrennt

Nachdem Sie nun mit den wichtigsten Dingen vertraut sind, können wir nochmals die Frage aufgreifen, ob es, wenn man ohnehin mehrere Geräte erwerben will, sinnvoll ist, diese in Form eines Doppellaufwerks zu kaufen. Zunächst einmal spricht für diese Lösung der Preis. Doppellaufwerke sind aus Gründen der niedrigeren Produktionskosten (und natürlich auch aus Gründen der Werbung) billiger als zwei einzelne Geräte. Wunderbar - bei gleicher Leistung ein paar Scheine gespart. Nicht ganz: Doppellaufwerke bringen auch Nachteile mit sich: Tritt z.B. bei einem der Laufwerke ein Defekt auf, muß man das komplette Gerät (also auch das intakte Laufwerk) zur Herstellerfirma schicken - und somit mit einem einzigen Laufwerk auskommen, was, wenn man schon einmal mit mehreren Geräten gearbeitet hat, ziemlich nervenaufreibend ist. Der andere Nachteil liegt in der fehlenden Möglichkeit des getrennten Weiterverkaufs. Nach soviel Nachteilen sind wir den Doppelfloppies noch ein paar Pro-Argumente schuldig. So rutschen sie z.B. nicht so leicht nach hinten, wie es einzelne 3½-Zoll-Laufwerke tun, wenn man die [Eject]-Taste drückt. Auch kommt man mit weniger Kabeln und etwas weniger Platz aus.

4.3 Festplatten am Amiga 500

Auch wenn wir befürchten müssen, uns zu wiederholen, wollen wir es nochmals betonen: Der Amiga lädt viel von seinen Diskettenlaufwerken nach. Mit Hilfe diverser Tricks (z.B. der RAM-Disk) läßt sich zwar ein Teil der Zugriffe beschleunigen, wobei dann allerdings gewisse Abstriche (z.B. weniger Speicher) zu machen sind. Größere Programme muß man nach wie vor von Diskette laden. Daß das nicht gerade schnell geht, zeigt das Wölkchen, in das sich der Mauszeiger verwandelt, wenn man ein Programm über die Workbench aufruft. Im Vergleich zur Lade-Geschwindigkeit anderer Computer sind die normalen Amiga-Laufwerke zwar sehr schnell; betrachtet man jedoch die Menge der Daten und die Länge von Amiga-Programmen, wird man feststellen, daß eine erheblich schnellere "Diskette" auf jeden Fall wünschenswert wäre. Auch die Ausgabe des Directories einer AmigaDOS-Diskette kann viel Zeit verschlingen, weil die einzelnen Blöcke der Diskette, die deren Inhaltsverzeichnis enthalten, über alle Spuren verstreut liegen.

Nun gibt es aber ein Gerät mit dem schönen Namen Harddisk. Eine solche Harddisk - manche sagen auch Festplatte oder einfach nur Platte zu ihr - ist ein riesiges Diskettenlaufwerk, dessen Datenträger (die Magnetscheibe) nicht gewechselt werden kann. Die Platte, auf der alle Informationen gespeichert sind, ist fest in ein Gehäuse eingebaut - daher auch der Name: Festplatte.

Normale Harddisks erreichen Speicherkapazitäten von 20, 40 oder noch mehr MByte. Doch nicht nur die Kapazität ist entscheidend, sondern auch die Geschwindigkeit überzeugt. Eine Harddisk, die mit dem ab Workbench 1.3 verfügbaren FastFileSystem formatiert wurde, kann sich mit der RAM-Disk messen!

Gründe, die für eine Festplatte sprechen:

- Häufig benutzte Programme können schnell geladen werden, ohne daß extra eine Diskette gesucht und gebootet werden müßte.

- Ständig (also für jedes aufgerufene Programm) sind alle Zeichensätze, CLI-Befehle, Libraries etc. verfügbar, da sich diese auf der Platte unterbringen lassen.

- Der Rechner kann (jedenfalls bei einer bootfähigen Festplatte - z.B. Vortex) ohne Startdiskette "hochgefahren" (gebootet) werden. Daraus ergibt sich ein riesiger Zeit-Vorteil.

- Mehrere Programme lassen sich problemlos nacheinander aufrufen, ohne daß, wie es leider beim reinen Disketten-Betrieb häufig zur Gewohnheit wird, mit jedem Programm eine andere Diskette gebootet wird und damit der Inhalt der RAM-Disk samt aller Einstellungen verloren geht.

- Software läßt sich erheblich schneller entwickeln, da Include-Files (s. C-Kapitel) und sonstige Daten enorm schnell verfügbar sind.

Für wen sich eine Festplatte nicht lohnt

Nicht für jeden Amiga-Anwender bringt eine Festplatte Vorteile in einem Umfang mit sich, die die Anschaffungskosten rechtfertigt. Eine Festplatte ist nämlich nur dann wirklich sinnvoll, wenn häufig ganz bestimmte Anwen-

dungsprogramme (Text-, Datei-Verwaltung, Programmentwicklungssysteme usw.) verwendet werden. Wenn Sie Ihre Programme nur selten einsetzen, keine Daten zwischen verschiedenen Systemen austauschen und nicht unbedingt darauf angewiesen sind, bestimmte Programme und Daten ständig verfügbar zu haben, sind Sie mit einem zweiten Laufwerk auch ganz gut bedient.

Aufbau eines Harddisk-Systems

Zum Betrieb einer Platte sind neben der nötigen Treiber-Software zwei Dinge erforderlich: das eigentliche Aufzeichnungsgerät (Plattenlaufwerk) und die Steuerelektronik (der Controller). Man trennt diese beiden Komponenten aus einem einfachen Grund: Möchte man mehrere Platten anschließen, kann man diese an einem einzigen Controller betreiben. Festplatten, die man für den Amiga 500 kauft, werden meistens in einem Gehäuse geliefert, das sowohl die Platte als auch den Controller enthält. Bei einigen Fabrikaten kann man noch eine zweite Platte in das Gehäuse einbauen oder daran anschließen, ohne einen weiteren Controller zu installieren. Bei anderen Fabrikaten sind Platte und Controller in verschiedenen Gehäusen untergebracht, die mit einem Flachband- oder Rundkabel miteinander verbunden werden. Diese Methode hat noch einen weiteren Vorteil: man kann die Platte auf ein Wandregal oder einen anderen Tisch stellen, auf dem sie während des Betriebs noch ruhiger steht.

Moderne Festplatten sind zwar so sicher gebaut, daß sie auch während des Betriebs kleine Erschütterungen überstehen, doch sollte man hier lieber kein Risiko eingehen. Weiterhin kann ein zu lauter Lüfter das Interesse daran steigern, die Platte mit einer gewissen Distanz zum Arbeitsplatz aufzustellen. Computer und deren Zubehör werden nämlich in der Regel mit einem internen Lüfter ausgestattet, der die während des Betriebs anfallende Wärme ableiten soll, um zu verhindern, daß sich die Chips zu stark erhitzen, was ihre Lebenserwartung stark reduzieren könnte. Besitzer der 1541-Floppy für den Commodore 64 können von dieser Problematik ein Liedchen singen.

Im Amiga 500 selbst ist kein Lüfter eingebaut. Er kommt aber nur deshalb ohne einen solchen aus, weil sein Netzteil in einem separaten Gehäuse untergebracht ist. Wie groß die Hitzeentwicklung wäre, wenn dieses Netzteil im Amiga selbst integriert wäre, können Sie feststellen, wenn Sie die Temperatur des Netzteiles nach mehreren Betriebsstunden mit der Hand messen!

Wenn Sie im Besitz des Bausatzes aus Kapitel 4.7 sind und damit Ihrem 500er einige 2000er-Steckplätze verpaßt haben, können Sie auch ein Platten-System verwenden, das für den Amiga 2000 entwickelt wurde.

Dabei gibt es verschiedene Möglichkeiten:

1) Steht für Sie fest, daß Sie nur eine Platte installieren möchten und auch keine PC-Karte kaufen werden, um auch MS-DOS-Software verwenden zu können, ist es ratsam, eine sogenannte File-Card zu verwenden. Bei diesem System sind auf einer Steckkarte (in diesem Fall einer 2000er-Karte) sowohl der Controller als auch eine Platte installiert. File-Cards kann man schon für ca. 600-1000 DM erwerben.

2) Wenn Sie in Erwägung ziehen, später eine weitere Platte zu installieren, und keine PC-Karte kaufen möchten, können Sie die von Commodore hergestellte SCSI-Karte (A2090a) verwenden, an die Sie dann zwei gewöhnliche und acht SCSI-Platten-Laufwerke anschließen können. Die SCSI-Schnittstelle hat einen entscheidenden Vorteil: Über sie können nicht nur Festplatten, sondern auch andere Geräte angeschlossen werden. Voraussetzung dafür ist allerdings ein passender Software-Treiber. Über SCSI kann man z.B. optische Speicher bzw. CD-ROM betreiben. Schafft man sich eine SCSI-Karte an, hat man also schon einen Schritt in die Zukunft getan.

3) Möchte man sich eine PC-Karte anschaffen, sollte man in Erwägung ziehen, eine File-Card auf der PC-Seite zu installieren. Auf ihr kann man dann auch eine Partition für den Amiga einrichten. Der PC kann von seiner Platte sogar booten. Nähere Informationen zu diesem Thema können Sie dem großen Amiga-2000-Buch entnehmen. Um eine PC-Karte an den Amiga anzuschließen, müssen Sie ja ohnehin den Bausatz der Firma Comptec, den wir Ihnen in Kapitel 4.7 vorstellen, verwenden, der den Amiga zu einem dreiviertel 2000er aufrüstet.

4) Auch auf der PC-Seite kann man Controller und Platte(n) getrennt einbauen. Der Controller belegt dann einen PC-Steckplatz. Als Plattenlaufwerk verwendet man im allgemeinen ein 3½-Zoll-Gerät, das an die Stelle des zweiten internen Laufwerks eingebaut wird.

Die oben genannten Möglichkeiten lassen sich sicher noch variieren und kombinieren. Ohne den angesprochenen Bausatz muß man ohnehin eine der externen Platten verwenden.

Sinnvoller Einsatz der Platte

Die schönste und größte Platte nützt nichts, wenn man sie nicht effektiv und vor allem sinnvoll einzusetzen versteht. Im allgemeinen ist es beispielsweise nicht ratsam, die PD- oder Spiele-Sammlung auf der Platte unterzubringen. Sinnvoller sind da schon CLI-Befehle und Anwendungsprogramme bzw. deren Daten-Files. Allerdings sollte man seine Daten nach einem gewissem System auf der Platte anordnen, um Datenverlust zu vermeiden und immer den Überblick zu behalten. Auch empfiehlt es sich, ab und zu die Files, die man noch nicht in der neuesten Form auf einer Diskette gesichert hat, zu kopieren.

Uff! - Ganz schön viel. Wir sind uns der Komplexität der Materie wohl bewußt und gehen daher ab jetzt ganz langsam vor. Beginnen wollen wir mit der Verplanung der Platte. Dazu sind einige Vorkehrungen zu treffen, die später nicht ohne weiteres geändert werden können.

Im allgemeinen werden Amiga-Festplatten bereits formatiert (und damit auch partitioniert) ausgeliefert. Wer sich mit der Einteilung zufrieden gibt oder sich nicht traut, die Platte komplett zu löschen, um eigene Strukturen zu realisieren, kann die Einstellung auch unverändert lassen.

Eine Festplatte kann in mehrere Partitionen unterteilt werden, die von AmigaDOS wie völlig unabhängige Geräte angesprochen werden können. Von der Möglichkeit der Partitionierung sollte man Gebrauch machen, da man dann z.B. die Möglichkeit hat, eine einzelne Partition und damit nur einen Teil der Platte zu formatieren.

Sinnvoll ist es z.B., eine eigene Partition für die Workbench zu schaffen. Wie man diese dann sinnvoll einsetzt, schildern wir später noch. Diese Partition sollte man nicht zu klein ansetzen. Zwei MByte müßten jedoch ausreichen. (Ein MByte für die Workbench und ein weiteres für Programme, die man eigentlich schon immer auf die Workbench kopieren wollte, aber nicht konnte, da dort nicht genügend Platz war.)

In einer weiteren Partition kann man dann Programme und Daten unterbringen. Unter Umständen kann man Daten und Programme auch auf verschiedene Partitionen verteilen. Dies sollte man allerdings nur dann tun, wenn man schon ganz genau weiß, wieviel Platz man für die Programme benötigt - also fast nie.

Sehr wichtig ist es, darauf zu achten, daß keine (oder wenigstens möglichst wenige) Programme und Daten im Hauptverzeichnis einer Platten-Partition liegen. Alles sollte in Unterverzeichnissen abgelegt werden. Diese kann man dann von der Workbench einfach auf eine Diskette umkopieren. Voraussetzung dafür ist allerdings, daß der Inhalt jeder Schublade maximal 880 KBytes hat. Natürlich gibt es auch zu dieser Regel einer Ausnahme: Programme, die ein File über

```
:Beispiel-File
```

nachladen, können die gute Absicht vereiteln. Wenn es keine Möglichkeit gibt, den Zugriffsnamen zu ändern (hier wäre es z.B. möglich, festzustellen, ob sich mittels Default Tool oder Tool Types etwas machen läßt), muß man das File (in unserem Fall heißt es Beispiel-File) im Hauptverzeichnis ablegen. Sichert man die Platte auf Disketten, muß man unbedingt auch die Files sichern, die man nicht in Schubladen deponiert hat.

Wenn man die Platte komplett eingerichtet hat, sollte man jede Schublade auf eine Diskette kopieren!!!

Sollte es einmal nötig werden, die Platte neu zu formatieren, weil sich ein Read-Error eingeschlichen hat, kann man die Sicherheitskopien einfach wieder zurückkopieren. Wichtige Daten, die man auf der Workbench ablegt, sollte man ohnehin möglichst oft auf eine normale Diskette kopieren, da sie auf der Festplatte nicht ganz so sicher sind.

Diesen Rat sollten Sie sehr ernst nehmen, da es ein ziemlich unangenehmes Gefühl ist, wenn man ein 300 KByte langes Assembler-Quellcode-File wegen eines Read-Errors auf der Platte verliert. Nur wenn man mit den richtigen Werkzeugen ausgerüstet ist, kann man hier noch etwas retten. Wenn man bedenkt, daß eine Festplatte über 600 Spuren hat, kann die Suche nach Daten-Blöcken aber trotzdem sehr langwierig werden.

Hat man die Partition, in der die Panne passiert ist, normal formatiert, kann man zum Retten der Daten oft den DiskDoctor einsetzen.

Formatierung der Harddisk

Amiga-Festplatten werden im allgemeinen schon formatiert geliefert. Wenn dies nicht der Fall ist oder wenn es nötig sein sollte, die Platte neu zu formatieren, wird es schon kompliziert. Festplatten werden nämlich nicht wie normale Disketten formatiert, sondern müssen erst vor- und dann richtig formatiert werden. Im Gegensatz zu Vor- und Hauptwäsche muß die Vorformatierung, wenn es nicht schon geschehen ist, mindestens einmal durchgeführt werden. Die eigentliche Formatierung wird dann mit dem Format-Befehl der Workbench vorgenommen.

Da die Programme zum Vorformatieren einer Festplatte von Hersteller zu Hersteller anders aussehen, können wir Ihnen leider keine direkten Anleitungen für diesen Vorgang an die Hand geben. Die Handbücher (oder Handhefte) der Festplatten enthalten jedoch Informationen zu diesem Thema.

Die Nachformatierung der Partitionen erfolgt mit Format oder unter Umständen mit DPFormat. Normalerweise funktioniert jedoch Format.

```
Format DRIVE DH0: NAME Platte0
```

würde z.B. die Partition DH0: formatieren und ihr den Namen Platte0 geben. Sie können auch das neue, schnellere FastFileSystem verwenden. Eine Partition, die unter diesem System betrieben wird, kann z.B. so formatiert werden:

```
Format DRIVE DH0: NAME Platte0 FFS
```

Das Kürzel FFS signalisiert, daß Sie die Verwendung des FastFileSystems wünschen.

Bevor eine Platte formatiert werden kann, muß sie gemounted werden. Dazu dient der CLI-Befehl Mount, der die Datei Devs:MountList liest und ihr die Parameter für die Plattenpartition entnimmt. In der StartUp-Sequence einer Boot-Diskette, die die Harddisk aktivieren soll, muß für jede Partition ein Mount-Befehl stehen. Zunächst sehen wir uns jedoch die Einträge in der MountList an. Dazu geben wir im CLI den Befehl

```
Ed DEVS:MountList
```

ein. Für jede Partition legen wir nun einen Eintrag an. Auf den System-Disketten, die mit einer Harddisk geliefert werden (sollten), befindet sich eigentlich immer eine Beispiel-MountList, die man oft direkt verwenden kann.

Dennoch wollen wir Ihnen den Aufbau eines MountList-Eintrags beschreiben.

Jeder Eintrag beginnt mit dem Namen des Geräts - in unserem Beispiel DH0:. Danach kommen die Daten. Beendet wird er durch das Zeichen #.

Die Einträge in der MountList, die wir für die Festplatte von CombiTec verwendet haben, sahen wie folgt aus:

```
DH1:    Device = harddisk.device       /* Name: Device-Treibers  */
        FileSystem = 1:FastFileSystem  /* Name: File-Systems     */
        Unit   = 0                     /* Phys. Laufwerksnummer  */
        Flags  = 0                     /* Flags für OpenDevice() */
        Surfaces  = 4                  /* Köpfe                  */
        BlocksPerTrack = 17            /* Sektoren pro Spur      */
        Reserved = 2                   /* DOS-Erkennungsblöcke   */
        Interleave = 0                 /* Interleave des DOS     */
        LowCyl = 0                     /* ... dieser Partition   */
        HighCyl = 25                   /* ... dieser Partition   */
        Buffers = 30                   /* zwischengepeicherte    */
                                       /*                 Blöcke */
        BufMemType = 1                 /* Fast oder Chip-Ram     */
        GlobVec = -1                   /* FFS: -1, sonst: 0      */
        DosType = 0x444F5301           /* Kennzeichen für FFS    */
        Mount = 1                      /* FFS + Treiber          */
                                       /*           gleich laden */
        StackSize = 8192               /* Stack für FFS-Process  */
        Priority = 5                   /* Priorität für          */
                                       /*        FFS-Process     */
#

DH0:    Device = harddisk.device
        FileSystem = 1:FastFileSystem
        Unit = 0
        Flags = 0
        Surfaces = 4
        BlocksPerTrack = 17
        Reserved = 2
        Interleave = 0
        LowCyl = 26
```

```
        HighCyl = 614
        Buffers = 30
        BufMemType = 1
        GlobVec = -1
        DosType = 0x444F5301
        Mount = 1
        StackSize = 8192
        Priority = 5
#
```

Dabei ist DH1: die Workbench-Partition. Warum wir die Workbench nicht DH0: genannt haben, hat einen einfachen Grund. Viele Programme können nämlich nur die DH0:-Partition über ein Gadget wählen. Da man aber meistens mit der Daten- und Programm-Partition arbeitet, haben wir ihr den Namen DH0: gegönnt und die Workbench auf DH1: verbannt.

Zugegeben: die MountList ist nicht ganz einfach zu verstehen. Wir werden Ihnen daher die Funktion jedes einzelnen Eintrages beschreiben.

Device

Über Device legen wir fest, welcher Treiber für die Platte zuständig ist. In unserem Fall war dies harddisk.device. Andere Platten arbeiten z.B. mit omti.device und hddisk.device. Den entsprechenden Treiber müssen Sie im allgemeinen von der System-Diskette der Festplatte ins Devs-Verzeichnis der Festplatten-Startdiskette kopieren. Einige Firmen liefern sogar schon fertige Workbench-Disketten mit ihren Produkten aus. Allerdings sind diese nicht immer so eingerichtet, wie man es gerne hätte.

FileSystem

Um das FastFileSystem zu verwenden, muß man den Parameter FileSystem auf L:FastFileSystem zeigen lassen. Allerdings ist dies erst ab Workbench 1.3 möglich. Achten Sie darauf, daß sich die Datei FastFileSystem auch im L:-Verzeichnis befindet.

Unit

Welches physikalisches Gerät angesprochen werden soll, legen Sie mit Unit fest. Im allgemeinen steht hier 0. Ein weiteres Plattenlaufwerk würde die Nummer Eins erhalten. Etwas anders sieht es auch aus, wenn man den SCSI-Controller verwendet. Man muß dann dem ersten angeschlossenen Gerät die Unit-Nummer Eins und dem zweiten die Zwei geben. SCSI-Platten werden beginnend mit Unit drei durchnumeriert. (Jedenfalls gilt das für den A2090a-Controller von Commodore.)

Flags

Die Funktion von Flags hängt vom verwendeten Device ab. Normalerweise setzt man Flags auf Null. Nähere Informationen entnehmen Sie bitte der Anleitung Ihrer Platte bzw. Ihres Treibers.

Surfaces

Mit Surfaces stellen Sie ein, wie viele Oberflächen (und damit Schreib-/Leseköpfe die Platte hat. Dieser Wert ist im allgemeinen doppelt so hoch wie die Anzahl der Magnetplatten. Eine 20-MByte-Harddisk hat normalerweise zwei solcher Platten und damit vier Schreib-/Leseköpfe.

BlocksPerTrack

BlocksPerTrack hängt auch wieder vom Modell der Platte ab. Hier muß man angeben, wie viele Blöcke sich auf jeder Spur befinden. 20er-Festplatten verwenden üblicherweise 17. Der Anleitung Ihrer Festplatte müßten Sie jedoch den richtigen Wert entnehmen können. Zur Not fragen Sie einfach einen Fachhändler.

Reserved

Hiermit legen Sie fest, wie viele Boot-Blöcke reserviert werden sollen. Gewöhnlich wählt man zwei.

Interleave

Auch dieser Parameter kann, ähnlich wie Flags von Device zu Device unterschiedliche Funktionen erfüllen. Eigentlich ist er dafür vorgesehen, den Interleave-Faktor der Platte einzustellen, von dem es unter anderem abhängt, wie schnell diese ist. Der Wert bestimmt sozusagen, mit welchem Abstand die Blöcke auf der Platte eingerichtet werden. Ist dieser Wert kleiner als für die Festplatte empfohlen wurde, muß sich die Magnetscheibe für jeden zu lesenden Block einmal komplett drehen, was naturgemäß zu Einbußen bei der Geschwindigkeit führt. Deshalb sollte Interleave auch nicht geändert werden.

LowCyl

Wie Sie bereits wissen, kann man eine Festplatte in mehrere Partitionen aufteilen, die dann (wenigstens auf File-Ebene) wie völlig eigenständige Geräte angesprochen werden können. Mit LowCyl legt man fest, bei welcher Spur eine Partition beginnt.

HighCyl

Es dürfte nicht schwer sein, zu erraten, daß HighCyl das Gegenstück zu LowCyl ist und somit die letzte Spur einer Partition bezeichnet.

Buffers

Dieser Parameter dient dem Zweck, die Anzahl der Puffer festzulegen, in denen einzelne Blöcke zwischengespeichert werden können. Je höher dieser Wert ist, desto schneller arbeitet das Gerät. Man sollte es aber nicht übertreiben, da der Verwaltungsaufwand für die zusätzlichen Speicherbereiche den

Geschwindigkeitsvorteil abbaut, wenn derer zuviele vorhanden sind. Festplatten sind in der Regel ohnehin schon so schnell, daß es gar nicht mehr nötig ist, sie noch schneller zu machen - besonders, wenn man berücksichtigt, daß dabei kostbarer RAM-Speicher verbraucht wird.

BufMemTyp

Hiermit stellen Sie ein, in welchem Speicher die Puffer eingerichtet werden sollen. 0 und 1 lassen sowohl Chip- als auch Fast-RAM zu. Das Fast-RAM wird allerdings, wenn noch genügend davon vorhanden ist, bevorzugt. 2 und 3 lassen nur Chip-RAM zu. Verwendet man 4 oder 5, wird nur Fast-RAM belegt. Um zu vermeiden, daß Fehler auf Systemen auftreten, die kein Fast-RAM haben, sollte man statt 4 und 5 lieber 0 und 1 einsetzen.

GlobVec

Verwendet man das FastFileSystem, muß man diesen Parameter auf -1 setzen. Andernfalls stellt man 0 ein.

DosType

DosType wird vom Mount-Befehl anerkannt. Über diesen Parameter legt man die Kennung des verwendeten FileSystems fest. Diesen Parameter muß man nur angeben, wenn man nicht das normale AmigaDOS-, sondern ein anderes Aufzeichnungsformat wählt (z.B. FFS = FastFileSystem). Im Falle des FFS verwendet man 0x444F5301. Warum? Die erste vier gibt die Anzahl der folgenden Kennungsbytes an. Es sind $44, $4F, $53 und $01 ("DOS"+CHR$(1)). Die normale DOS-Kennung unterscheidet sich nur im letzten Byte, in dem keine (binäre) Eins, sondern eine Null steht.

Mount

Mittels dieses Parameters legen Sie fest, ob die Gerätetreiber sofort oder erst beim ersten Zugriff auf das neue Gerät geladen werden sollen. In unserem

Beispiel werden sie sofort eingebunden. Verwendet man -1, werden die Treiber erst geladen, wenn das erste Mal auf das Gerät zugegriffen wird.

StackSize

Gibt an, wie groß der Speicherbereich sein soll, den der Harddisk-Treiber verwenden kann, um Werte schnell auf dem Stack zwischenzuspeichern. Diesen Wert sollten Sie, falls er vom Hersteller vorgegeben ist, nicht herabsetzen, da ein zu kleiner Stack einen System-Absturz oder stillen Datenverlust nach sich ziehen kann. Was es mit dem Stack genau auf sich hat, erfahren Sie vielleicht schon bald - nämlich dann, wenn Sie das Assembler-Kapitel lesen.

Priority

Mit diesem Parameter kann man die Rangordnung des Treibers relativ zu anderen Tasks einstellen. Sie können hier auch ganz neutral Null einsetzen.

Eine günstige StartUp-Sequence

Solange es noch nicht ohne weiteres möglich ist, die Workbench komplett von der Platte zu booten, muß man sich Gedanken darüber machen, wie man die StartUp-Sequence der Start-Diskette am günstigsten einrichtet.

Man sollte versuchen, so früh wie möglich alle Zugriffe auf die Platte zu verlegen. Das folgende Beispiel geht davon aus, daß in DH1: die normalen Workbench-Files zu finden sind. Auf der Boot-Diskette muß sich die MountList befinden.

StartUp-Sequence der Diskette

```
Mount DH1:
Mount DH0:
DH1:c/Assign SYS: DH1:
DH1:c/Assign C: DH1:c
DH1:c/Assign S: DH1:s
DH1:c/Assign L: DH1:l
DH1:c/Assign FONTS: DH1:fonts
```

```
DH1:c/Assign DEVS: DH1:devs
DH1:c/Assign LIBS: DH1:libs
CD DH1:
Execute S:Startup-Sequence
```

StartUp-Sequence in DH1:s/StartUp-Sequence

```
System/SetMap d
AddBuffers df0: 20
AddBuffers df1: 20
Stack 20000
BindDrivers
SetClock Opt Load
Resident CLI L:Shell-Seg SYSTEM pure add
Resident c:Execute pure
Mount NewCon:
FailAt 11
Path RAM: SYS:Utilities SYS:System
LoadWB
EndCLI >NIL:
```

Produktvorstellung

Immer mehr Hersteller bieten immer mehr Festplatten für den Amiga an. Wir haben einige davon herausgepickt, um sie Ihnen vorzustellen.

Combitec

Beginnen wir mit der Combitec HD 20. Diese Platte wird in einem soliden Gehäuse geliefert, das über ein Flachbandkabel mit einem Adapter verbunden wird, den man an den Expansionsport (links am Amiga 500) anschließt. Besitzt man eine RAM-Karte der Firma Combitec, braucht man diesen Adapter nicht zu kaufen, da die RAM-Erweiterungen (am Expansionsport) eine Anschlußmöglichkeit für die Platte bieten.

Eingeschaltet wird die Harddisk über zwei Schalter. Einer davon befindet sich am Hauptgehäuse und ein weiterer, der sogenannte Konfigurationsschalter, am Adapter.

Die Festplatte arbeitete während der Testphase einwandfrei. Es stellten sich auch bei Erschütterungen des Tisches, auf dem wir sie abgestellt hatten, keine Read-Errors ein.

Bedauerlicherweise werden die Schreib-/Leseköpfe der Platte nicht automatisch in eine Position gebracht, in der sie auch bei starken Erschütterungen die Oberflächen der Magnetscheiben nicht berühren. Auf der mitgelieferten System-Diskette befindet sich jedoch ein Programm, das diese Aufgabe übernimmt. Das Fehlen einer Auto-Park-Funktion wirkt sich unter Umständen, wenn die Festplatte stärkeren Bewegungen ausgesetzt ist, negativ auf die Haltbarkeit aus, d.h. wenn man etwas Pech hat, können Teile der Magnetscheibe zerstört werden, wenn einer der Leseköpfe die Scheiben berührt. Man nennt dieses unangenehme Ereignis übrigens Head-Crash.

Auf der Combitec-Platte findet der Käufer den Inhalt acht ausgewählter Public-Domain-Disketten der Fred-Fish-Serie.

Message

Die Festplatten der Firma Message zeichnen sich besonders durch viele mögliche Extras aus. So kann man z.B. mit einem kleinen Kabel die Stromversorgung des 500ers über das Netzteil der Platte abwickeln. Dies hat zwei Vorteile: Erstens kann man mit einem einzigen Schalter beide Geräte anschalten, zweitens kann man nun an den Amiga-Ports mehr Verbraucher (z.B. Laufwerke) anschließen, ohne das unterdimensionierte Amiga-Netzteil zu überlasten.

Wer auf den Geschmack gekommen ist, kann im gleichen Gehäuse eine weitere Festplatte nachrüsten, ohne einen weiteren Controller kaufen zu müssen. Auf diese Weise spart man Geld und Platz.

Auch eine Auto-Park-Funktion wurde in die extra für die Platte entwickelte Software eingebaut. Greift man einige Sekunden nicht auf die Platte zu, werden alle Schreib-/Leseköpfe geparkt, um mögliche Risiken einer Beschädigung zu vermeiden.

Golem

Preiswert und bedienerfreundlich ist auch die Golem-Festplatte, die über die Firma Kupke bezogen werden kann. Die Festplatte ist in verschiedenen Ausführungen (von 20 bis 60 MByte) lieferbar. An ihr werden wir exemplarisch zeigen, wie einige knifflige Dinge mit Festplatten ausgeführt werden.

Lieferumfang (HD 3000 mit 20 MByte für Amiga 500)

Neben der eigentlichen Festplatte, die in einem flachen Metallgehäuse untergebracht ist, erhält der Käufer einen Anschluß-Adapter für den Amiga 500. Weiterhin liegt dem Paket eine 12seitige Anleitung und eine bootfähige System-Diskette bei, auf der sich auch ein Programm befindet, mit dem die Platte komfortabel verwaltet werden kann. Die HD 3000 wird formatiert ausgeliefert, wobei eine Aufteilung in zwei gleichgroße Partitionen vorgenommen wurde. Allerdings arbeitet nur eine dieser Partitionen mit dem Fast-File-System.

Der Anschluß

Die Installation der Hardware gestaltet sich denkbar einfach. Zunächst wird die eigentliche Platte in der Nähe des Rechners aufgestellt. Das Verbindungskabel, mit dem die Verbindung zum Adapter hergestellt wird, ist ziemlich knapp bemessen. Wegen der geringen Höhe des Festplattengehäuses ist dies aber nicht weiter schlimm, da auf ihm diverse andere Gegenstände abgelegt werden können. Die Festplatte wird mit einem normalen Kabel mit einer Steckdose verbunden. Nun muß der Adapter an den Expansionsport des 500er angeschlossen werden. Bei uns ergab sich dabei ein kleines Problem: Die Buchsenleiste des Adapters war zu hoch angebracht, so daß sie nicht einfach von der Seite hereinzuschieben war. Diese Panne konnten wir beheben, indem wir den Rechner auf eine Schreibtischunterlage stellten, so daß er einige Millimeter höher stand. Möglicherweise tritt dieses Problem nicht bei allen Amiga 500 auf. Commodore hat die Produktion des 500ers nämlich schon mehrmals geändert, und wird es voraussichtlich auch noch einige Male tun. Ist der Adapter montiert, wird das Verbindugskabel der Platte eingesteckt. Das System ist jetzt betriebsbereit.

Die Software

Sehr gut gefallen hat uns die Software zur Golem-Festplatte. Sie ist sehr übersichtlich aufgebaut und besonders leicht zu bedienen. Auch die zahlreichen zusätzlichen Möglichkeiten verdienen Beachtung. Die mitgelieferte Diskette ist von Anfang an zum Booten geeignet, doch sollte man möglichst schnell eine eigene Start-Diskette anlegen. Wie das geht, zeigen wir noch.

Schalten Sie jetzt bitte die Festplatte und den Amiga an. Dann booten Sie die Golem-Start-Diskette. Nach einiger Zeit erscheint die Workbench-Oberfläche, auf der zwei neue Icons zu sehen sind, die die Plattenpartitionen vertreten.

Wir laden jedoch das Hauptverzeichnis der Boot-Diskette mit einem Doppelklick. In dem Fenster, das sich nun öffnet, finden wir neben einigen unwichtigen Dingen eine Schublade mit dem Namen "Golem_HD". Wir öffnen sie. In ihr befindet sich das Programm "Formatmenue". Bevor wir es aufrufen, fertigen wir noch eine Sicherheitskopie der Diskette an, da das Formatierungsprogramm automatisch den Harddisk-Treiber auf der System-Diskette verändert.

Damit der Rechner die Kopie, die wir nur vorübergehend benutzen werden, auch anerkennt, booten wir sie. Dann rufen wir das CLI auf, um einen Blick auf die MountList zu werfen:

```
ED Devs:MountList
```

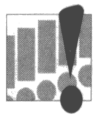
In der MountList wurden schon vier Einträge ergänzt: DH0:, DH1:, DH2:, und DH3:. Praktischerweise beziehen sich DH2: und DH3: auf eine zweite Platte, die nachgerüstet werden kann. Uns interessieren jedoch nur DH0: und DH1:, da diese Einträge die beiden Partitionen betreffen, die auf der Platte eingerichtet sind.

```
DH0:    Device = GOLEM.HDisk.device
        Unit   = 0
        Flags  = 0
        Surfaces  = 4
        BlocksPerTrack = 17
        Reserved = 2
        Interleave = 0
        LowCyl = 0
        HighCyl = 306
        Buffers = 30
```

```
        StackSize = 1000
        BufMemType = 1
        Mount = 1
#

DH1:    Device = GOLEM.HDisk.device
        Unit   = 0
        Flags  = 0
        Surfaces  = 4
        BlocksPerTrack = 17
        Reserved = 2
        Interleave = 0
        LowCyl = 307
        HighCyl = 614
        Buffers = 30
        StackSize = 1000
        BufMemType = 1
        Mount = 1
        GlobVec = -1
        MaxTransfer = 512
        DosType = 0x444F5301
        FileSystem = l:FastFileSystem
#
```

Entscheidend für uns sind nur die Werte LowCyl und HighCyl. Mit ihnen können wir die Größe der einzelnen Partitionen ändern.



Wenn Sie die Einstellung, mit der Ihre Platte geliefert wurde, nicht ändern wollen, sollten Sie die einzelnen Schritte nicht mitmachen, da Sie dadurch etwas verändern würden. Wer jedoch ein individuelles (sinnvolleres) Größenverhältnis der Partitionen zueinander einrichten will, erfährt im folgenden Abschnitt, wie er dies bewerkstelligen kann.

DH0: wird ohne und DH1: mit FFS formatiert. Bei Änderungen an Low- und HighCyl müssen wir beachten, daß eine Verschiebung der Anfangs- und Endwerte den Inhalt beider Partitionen zerstört. Normalerweise wird man die Werte so wählen, daß beide direkt aneinander liegen.

Zur Demonstration werden wir für DH0: die folgenden Werte einstellen:

```
LowCyl = 0
HighCyl = 99
```

Für DH1: wählen wir:

```
LowCyl = 100
HighCyl = 614
```

Um Mißverständnisse zu vermeiden, empfehlen wir, zunächst genau diese Einstellungen zu benutzen. Später können Sie natürlich vollkommen andere Werte wählen. Auch können Sie mehr als zwei Partitionen einrichten. Dabei muß darauf geachtet werden, daß keine Spur in zwei Partitionen gleichzeitig benutzt wird, d.h. es dürfen keine Überschneidungen bestehen. Auch darf HighCyl nicht größer sein als die Nummer der höchsten tatsächlich vorhandenen Spur.

Doch kommen wir zurück zu den von uns gewählten Werten. Zunächst speichern wir die Veränderungen, die wir an Low- und HighCyl vorgenommen haben mit "X". Dann legen wir die nicht schreibgeschützte Kopie der Golem-System-Diskette in DF0: ein und laden das Programm "Formatmenue" aus dem Ordner "Golem_HD" über die Workbench. Beachten Sie bitte, daß das Formatierungsprogramm immer annimmt, daß die Diskette in DF0: eingelegt ist. Andere Laufwerke werden nicht akzeptiert.

Das Menü des Programms bietet alles, was man sich wünschen kann. Zusätzlich zu den Gadgets kann man über die Menüleiste des Fensters, die beim Drücken der rechten Maustaste erscheint, wählen, welchen Plattentyp man formatieren will. Das Programm liest jedoch aus dem Harddisk-Treiber, den es aus dem DEVS-Verzeichnis der System-Diskette lädt, den Typ aus, mit dem die Diskette ausgeliefert wurde. In den meisten Fällen muß und darf man also nichts ändern.

Bevor wir weiterarbeiten, verschaffen wir uns einen Überblick über die Funktionen des Programms. Dabei beginnen wir mit den Gadgets, die eine Funktion auslösen.

Einige Gadgets bewirken, daß der Inhalt der kompletten Harddisk gelöscht wird. Da es keine Sicherheitsabfragen gibt, dürfen Sie auf keinen Fall ein Gadget anklicken, ohne sich genau zu überlegen, was Sie tun. Eine weitere Tücke des Programms ist, daß es, wenn es verlassen wird, versucht, alle Einstellungen in den Harddisk-Treiber zu schreiben, der in DF0:Devs liegt. Hat man also Änderungen vorgenommen, die nicht richtig sind (z.B. einen an-

deren Plattentyp gewählt), muß man die Diskette aus dem Laufwerk nehmen, bevor(!) man das Programm verläßt. Der Requester, der dann dazu auffordert, die Diskette wieder in Drive 0 einzulegen, muß in diesem Fall mit CANCEL beantwortet werden.

Format Hard

Klicken Sie dieses Gadget an, wird die komplette Platte vorformatiert. Sie müssen diese Funktion nur benutzen, wenn sich Read-Errors auf der Platte befinden. Sollten einige Spuren hardwaremäßig zerstört sein, sorgt Format Hard dafür, daß eine Reserve-Spur belegt wird, die dann vom Rechner aus über die Nummer der defekten Spur angesprochen werden kann. Eine fortlaufende Numerierung ist also gewährleistet.

Format Tracks Hard

Wenn Sie genau wissen, auf welcher Spur sich ein Fehler eingeschlichen hat, der sich durch erneutes Formatieren korrigieren läßt, können Sie diesen mit Format Tracks Hard beheben. Wir raten von dieser Methode jedoch ab und empfehlen, dann, wenn ein Fehler aufgetreten ist, alle Daten von der Platte auf Disketten umzukopieren und die Harddisk komplett neu zu formatieren.

Damit stellen Sie sicher, daß Sie nicht durch das Formatieren einer einzelnen Spur wichtige Steuerdaten gelöscht haben, die nötig sind, um die Verteilung der Files richtig zu verwalten. Das Fast-File-System ist für solche Fehler sehr anfällig, da sie nicht immer erkannt werden. Folge: langsam aber sicher werden einzelne Files teilweise überschrieben, ohne daß Sie etwas davon merken.

Verify HardDisk

Wird dieses Gadget angeklickt, überprüft das Programm alle Spuren der Festplatte auf Read-Errors - eine, wie wir meinen, sehr nützliche Einrichtung.

Format Soft

Nachdem die komplette Harddisk vorformatiert ist (Format Hard) muß jede einzelne Partition noch einmal mit dieser Funktion "überformatiert" werden. Dabei ist zu beachten, daß Format Soft nur für Partitionen ohne FFS verwendet werden kann. Bevor die Funktion aufgerufen wird, müssen noch Start- und Endspur der Partition eingestellt werden. Wir werden dies noch ausführlich anhand unserer Beispielwerte beschreiben.

Format F-F-S Soft

Diese Funktion entspricht dem vorgenannten Format Soft Gadget. Der einzige Unterschied liegt darin, daß hierbei FFS-Partitionen formatiert werden.

Format Tracks Soft

Mit dieser Funktion, von deren Verwendung wir auch abraten, können einzelne Tracks nachformatiert werden.

Disk-Format, Interleave, Steprate, Number of Heads, Number of Cylinders, StartHead und EndHead sind Werte, die das Programm automatisch einstellt. Sie geben die Eigenschaften der Platte wieder. Mit Unit-Number ist es möglich, auf die zweite physikalische Platte (nicht im Lieferumfang enthalten) zuzugreifen. All diese Werte sind für uns uninteressant. Lediglich StartTrack und EndTrack werden wir benutzen.

Mit diesen beiden Werten stellen wir, bevor wir Format Soft oder Format F-F-S Soft aufrufen, ein, welche Tracks bearbeitet werden sollen. Wir müssen hier immer die Werte einstellen, die wir in der MountList eingetragen haben.

Wir stellen also StartTrack auf 0000 und EndTrack auf 0099. Das Programm ist bei der Einstellung der Werte leider etwas eigenwillig. Die vier Plus- und Minus-Zeichen unter beiden Zahlen erhöhen oder erniedrigen die Stelle, unter der sie stehen, je um eine Zahl. Entsteht beim Anklicken von Plus bzw. Minus ein Übertrag, wird er in die nächste Stelle übernommen. Sehr ärgerlich ist das Verhalten bei Werten, die nicht im zugelassenen Bereich liegen. Dazu ein Beispiel: Die höchste zulässige Spur-Nummer bei einer 20-MByte-Festplatte

ist 614. Steht der Wert EndTrack auf 590, kann man nicht das Pluszeichen unter der 5 anklicken, da dadurch die Zahl auf 690 erhöht würde, was nicht zulässig ist. Unsere gute Absicht, aus der 90 eine 14 zu machen, kennt das Programm leider nicht. Am leichtesten hält man sich allen Ärger mit diesem System vom Hals, wenn man zuerst StartTrack und dann EndTrack auf 0000 setzt. Danach erhöht man zuerst EndTrack und dann StartTrack, wobei man immer mit der Stelle anfängt, die am weitesten rechts steht. Jetzt müßte es Ihnen möglich sein, 0000 und 0099 einzustellen.

Haben Sie das erledigt, können Sie Format Soft anklicken. Die Partition DH0: wird nun ohne Fast-File-System formatiert, was übrigens nur wenige Sekunden dauert. Die noch verbleibende Partition DH1: formatieren wir, indem wir StartTrack auf 0100 und EndTrack auf 0614 stellen und anschließend Format F-F-S Soft anklicken.

Wenn wir das Programm verlassen, schreibt es eventuell noch einige Daten in den Harddisk-Treiber. Das stört uns nicht weiter, da wir ja alles richtig eingestellt haben. Bevor wir die Festplatte einrichten können, müssen wir erst einmal booten, damit das Betriebssystem auch erfährt, daß ab jetzt andere Partitionen definiert sind. Doch, oh Schreck, bricht der Amiga den Boot-Vorgang mit der Fehlermeldung ab, daß er ein Programm nicht auf der Festplatte finden kann? Kein Wunder, wir haben sie ja auch formatiert und damit gelöscht.

Zunächst übertragen wir die Workbench-Diskette (und zwar die Original-Diskette von Commodore) auf die Festplatte in die Partition ohne FFS. Wir geben also ein:

```
Copy ?
```

Wenn AmigaDOS die Syntax des Befehls ausgegeben hat, legen wir die Workbench in DF0: und geben

```
DF0: DH0: ALL
```

ein. Dann können wir uns in aller Ruhe einen Tee aufbrühen und warten, bis die komplette Diskette übertragen ist.

Eigentlich könnten wir das Übertragen der Worbench auf die Harddisk auch mit dem Batch-Programm HDInstall aus dem Golem-Verzeichnis erreichen.

Dabei wird jedoch die Golem-Diskette übertragen. Das Verfahren, nach dem wir vorgehen, ist auch nicht viel aufwendiger und läßt auch andere Disketten zu.

Jetzt ist es an der Zeit, Festplatte und Workbench aufeinander abzustimmen. Wir fertigen eine Kopie der Original-Workbench-Diskette an, die wir mit Harddisk Startup oder etwas ähnlichem beschriften. Danach booten wir wieder die Kopie der Harddisk-Diskette, die mit einer Fehlermeldung abbricht. Die Startup-Sequence der 1.3er Kopie editieren wir nun. Und zwar soll sie folgendermaßen aussehen:

```
FailAt 10
Mount DH0:
Mount DH1:
CD DH0:
DH0:c/Execute DH0:s/Startup-Sequence
```

Im Golem_HD-Verzeichnis der Harddisk-Diskette befindet sich übrigens ein File Start, das genau diesen Inhalt hat und daher nur mit neuem Namen in das S-Verzeichnis unserer eigenen Boot-Diskette übertragen werden muß, z.B.:

```
Copy DF0:Golem_HD/Start RAM:
Copy C:Copy RAM:
```

Jetzt muß die anzulegende Boot-Diskette eingelegt werden:

```
RAM:Copy RAM:Start DF0:s/Startup-Sequence
```

Ebenfalls im Golem-Verzeichnis finden wir unter Startup-Sequence eine Datei, die nach DH0:s kopiert werden muß. Sie enthält erwartungsgemäß weitere Befehle, die beim Booten ausgeführt werden müssen.

```
Copy DF0:Golem_HD/Startup-Sequence DH0:s
```

Ganz wichtig sind auch die MountList und der Harddisk-Treiber. Beide kopieren wir zunächst von unserer behelfsmäßigen Kopie der Harddisk-System-Diskette in die RAM-Disk und von dort auf unsere neue Boot-Diskette.

```
Copy DF0:Devs/MountList RAM:
Copy DF0:Devs/GOLEM.HDisk.device RAM:
Copy C:Copy RAM:
```

Jetzt legen wir die neue Boot-Diskette ein:

```
RAM:Copy RAM:MountList DF0:Devs
RAM:Copy RAM:GOLEM.HDisk.device DF0:Devs
```

Besitzer eines Zweitlaufwerks hätten etwas eleganter kopieren können:

```
Copy DF0:Devs/MountList DF1:Devs
Copy DF0:Devs/GOLEM.HDisk.device DF1:Devs
```

Das war's. Sie können jetzt mit der neuen Diskette booten und, wenn Sie möchten, den Golem-Ordner über die Workbench in das Verzeichnis-Fenster von DH0: ziehen. Übrigens: Die Startup-Sequence in DH0:s kann noch erheblich verbessert werden. Es ist z.B. ratsam, alle Befehle einzubauen, die im normalen Start-File der Original-Workbench benutzt werden (Resident, Mount, ...).

Schreibschutz der Platte

Über einen Menüpunkt des Formatierungsprogramms können Sie einen mittels Software realisierten Schreibschutz für die Festplatte einstellen. Diese Information wird im Harddisk-Treiber direkt vermerkt. Nur, wenn Sie das Programm erneut laden, können Sie den Schutz wieder entfernen.

Parken der Platte

Bleibt noch das Park-Programm aus Golem_HD zu erwähnen. Es bringt die Leseköpfe einer Platte in eine Transport- oder Pausenstellung, um Schäden zu vermeiden, die sich bei Erschütterungen einstellen könnten. Das Programm wird über das CLI mit der Anweisung

```
Park 0
```

aufgerufen. Die Null gibt die Unit-Nummer der Platte an. Haben Sie also eine zweite Harddisk installiert, muß diese extra (mit Park 1) in den Ruhezustand versetzt werden. Nach dem Aufruf von Park wird die Kontrolle nicht an das CLI zurückgegeben. Es ist also nicht sinnvoll, Park während der Arbeit zu benutzen.

4.4 Soundsampling

Tolle Sachen kann er - unser 500er. In Spielen überzeugt er nicht nur mit toller Grafik und schneller Animation sondern auch mit guter Musik, realistischer Sprache und verblüffend echten Geräuschen. Autorennen werden nicht mehr durch ein simples Brummen begleitet. Jede Aktion wird mit einem echten Geräusch unterlegt. Das Einlegen eines Gangs, das Überdrehen des Motors und andere Dinge werden täuschend echt wiedergegeben.

Toll! Aber wie machen "die" das? Ganz einfach. Mit einem Kassettenrekorder bewaffnet, zieht man los und nimmt auf, was man über den Amiga wiedergeben möchte. Das kann eine Stimme, ein Geräusch oder auch ein Lied sein, das man unbedingt im Computer haben möchte. Hat man seinen Traumsound erst einmal auf Kassette, muß man ihn in irgend einer Weise in den Computer bekommen. Das Zauberwort heißt in hier Soundsampling oder Musikdigitalisierung.

Und was nicht unterschätzt werden sollte: Man muß nur wenig Geld ausgeben, um die nötige Hard- und Software zu erwerben.

Ein Ton ist aus einer Folge von Schwingungen zusammengesetzt. Die Frequenz der Schwingungen bestimmt die Tonhöhe. Je höher die Frequenz, desto höher der Ton. Ein "a" schwingt z.B. mit 440 Hertz. Hertz nennt man die Einheit, in der Frequenzen gemessen werden. Ein Hertz entspricht einer Schwingung pro Sekunde. Mit anderen Worten fällt und steigt die Spannung in einem Lautsprecher, der ein "a" von sich gibt, 440 mal in jeder Sekunde.

Und da liegt der Knackpunkt. Auf einem Notenblatt schreibt man die Note "a". Eventuell macht man noch einige Zusatzbemerkungen - dann ist aber auch schon Schluß. Bei Musikstücken, die man über den Amiga wiedergibt, verfährt man ähnlich. Man gibt eine Reihe von Tönen und Zusatzparametern ein. Das Musikprogramm wandelt diese dann in Zusammenarbeit mit dem Soundchip in Schwingungen um.

Ein digitalisierter Sound liegt schon in Form von Schwingungen vor. D.h. nicht die Töne, sondern die Schwingungen werden aufgezeichnet. Darin kann man sowohl einen Vor- als auch einen Nachteil sehen. Der Vorteil liegt darin, daß digitalisierte Musik (und besonders Sprache) viel genauer ist. Wo viel Licht ist, ist auch viel Schatten. Das ist bei Computern nicht anders. Hier

müßte man aber eher sagen: Wo viele Details sind, braucht man auch viel Speicher. Zu gut deutsch: Digitalisiertes nimmt viel Speicher in Anspruch.

Mit 512 KByte kann man aber schon einiges anfangen. Um endlich loslegen zu können, brauchen wir zunächst einen am Amiga 500 funktionsfähigen Soundsampler. Mittlerweile bietet fast jeder Hardware-Hersteller ein eigenes Gerät an. Dabei gibt es Preis- und Qualitätsunterschiede.

Unser Rat: Da die Unterschiede nicht besonders groß sind, sollten Sie lieber ein bißchen weniger Qualität für einen erheblich niedrigeren Preis in Kauf nehmen. Die vielen Optionen kann man mit der richtigen Software auch nachträglich erzeugen. Stereo, Hall...

Für die weitere Arbeit legen wir den Alcomp-Soundsampler für den Amiga 500 zugrunde. Dieses Gerät wird mit Anleitung und Public-Domain-Software für 79 DM verkauft. Ein Preis, den man annehmen kann. Wenn Sie ein billigeres Gerät finden, sind Sie damit wahrscheinlich auch ganz gut beraten.

Besonders wichtig ist die richtige Software. Ohne die läuft nämlich gar nichts. Mit dem Alcomp-Digitalisierer wird das Programm "Perfect Sound 2.0" ausgeliefert. Mit ihm kann man anfänglich gut arbeiten. Findet man jedoch seinen Spaß am Digitalisieren, sollte man sich nach besserer Software umsehen.

Zu klären bleibt noch, wie und wo der Sampler angeschlossen wird. Bevor wir ihn überhaupt anschließen können, müssen wir ihn erst kaufen. Hier ist jedoch Vorsicht geboten. Nicht jeder Amiga-Digitalisierer kann am 500er betrieben werden. Der Grund dafür ist, daß der Anschluß, über den das Gerät mit dem Amiga verbunden wird, geändert wurde. Man unterscheidet daher zwei Arten von Digitalisierern: die einen für die 1000er Amiga und die anderen für den Rest der Serie.

Alle Sampler haben eins gemeinsam: sie werden am parallelen Port angeschlossen, der meistens für Drucker verwendet wird. Für uns als 500er-Besitzer, hört es hier aber oft noch nicht auf. Wir müssen noch ein zweites Kabel anschließen. Bei unserem Gerät war das ein Stecker für den zweiten Gameport.

Irgendwie müssen die Töne in den Sampler gelangen. Hier können wir leider nicht vorhersagen, welchen Anschluß Ihr Gerät hat. Der Alcomp-Digitalisierer ist jedenfalls mit einer Cinch-Buchse ausgestattet. Wer den Computer

nicht zur Anlage oder die Anlage zum Computer bringen will, kann ein tragbares Kassettenradio oder einen Walkman verwenden. Meistens verfügen diese Geräte aber über keine Cinch-Ausgänge. Man muß sich hier mit einem Adapter helfen. Ein Tip am Rande: Sollten Sie eine Stereotonquelle verwenden, schalten Sie sie wenn möglich auf Mono um, oder verwenden Sie ein Kabel, das beide Kanäle zusammenführt. Sonst können sich nämlich unangenehme Effekte einschleichen.

Nun zur Software. Die verschiedensten Firmen bieten Programme an, mit denen man komfortabel digitalisieren und abspielen kann. Wer nicht unbedingt darauf angewiesen ist, mit dem Besten vom Besten zu arbeiten, kann sich auch an diverse PD-(= Public Domain)-Produkte halten. Sie sind erheblich billiger. Auch mit der Qualität kann man zufrieden sein. Allerdings stürzen PD-Programme öfters ab als kommerzielle.

Einige Digitalisierer werden von Haus aus mit Software ausgeliefert.

In diesem Buch werden wir ein Programm verwenden, das in der Ausgabe 6/87 des 68000er Magazins vorgestellt wurde, das damals noch keine reine ST-Zeitschrift war. Dieses Progamm kann man zusammen mit einigen anderen auf einer Diskette für ca. 30 DM bei Markt & Technik bestellen. Qualitativ ist dieses Software-System nicht gerade berauschend..., uns soll es aber genügen.

Das Programm öffnet ein Fenster, in dem vier gleichartige Bereiche gezeigt werden. Mit ihnen kann man zur gleichen Zeit vier verschiedene Stücke bearbeiten.

Um einen Sound zu laden, müssen wir über die Menüleiste die dafür zuständige Routine beachten. Drei Formate stehen zur Auswahl: IFF, Future-Sound und Data. Beginnen wir mit IFF. Sie haben diese Bezeichnung vielleicht schon in Verbindung mit Bildern gehört. Der IFF-Standard kann aber noch mehr. Hier dient er z.B. dazu, digitale Musik zu speichern. Im Gegensatz zu den anderen Formaten werden in einem IFF-File alle zum richtigen Abspielen nötigen Daten mitgespeichert. Das absolute Gegenbeispiel ist Data. Soundfiles dieses Formats enthalten nichts als die reinen Musikdaten. Eine gute Alternative zu IFF ist Future-Sound. Dieses Format speichert neben den eigentlichen Daten noch die Länge des Stücks und die Geschwindigkeit, mit der es abgespielt werden muß. Laden Sie ein fremdes File, versuchen Sie es zuerst mit IFF. Klappt das nicht, nehmen Sie Future-Sound. Werden vernünftige

Werte angezeigt, ist dieses Format richtig. Andernfalls nehmen Sie Data. Beim Abspeichern sollten Sie immer IFF oder Future-Sound verwenden, da so Ihre eingestellte Abspielgeschwindigkeit erhalten bleibt.

Zum Laden müssen Sie dem Programm mitteilen, welchen Sound Sie denn gerne hätten. Dazu öffnet es eine File-Request-Box. Mit den Feldern DF0, DF1 und DH0 stellen Sie ein, wo zuerst nach der Datei gesucht werden soll. DF0 und DF1 vertreten das interne und ein externes Laufwerk. DH0 ermöglicht es, auf die Festplatte zuzugreifen. Wollen Sie in ein Unterverzeichnis verzweigen, klicken Sie einfach dessen Namen an. In der Verzeichnishierarchie gelangen Sie wieder nach oben, indem Sie "parent (dir)" in der ersten Zeile anklicken. Haben Sie sich für einen Sound entschieden, klicken Sie diesen an und wählen anschließend LOAD. Möchten Sie doch nichts laden, können Sie den Requester mit CANCEL verlassen. Neben den Dateinamen ist ein Balken zu sehen. Über und unter diesem Balken wird ein Pfeil dargestellt. Mit diesen Pfeilen können Sie den Inhalt in der Anzeige verschieben, falls auf einer Diskette oder in einem Verzeichnis mehr Dateien sind, als auf einmal angezeigt werden können. Wenn Sie später eine eigene Produktion mit SAVE abspeichern wollen, können Sie den vorgesehenen Namen, wenn dieser noch nicht existiert, auch über die Tastatur eingeben. Klicken Sie dazu den Bereich an, in dem der Name angezeigt wurde, den Sie geladen haben. Nun erscheint ein Cursor. Sollte hier noch ein Text stehen, können Sie diesen mit [Del], [Backspace] oder [Amiga] [X] löschen. Dann geben Sie den Namen ein und wählen SAVE.

Nun werden wir gemeinsam einen Sound aufnehmen.

Zuerst müssen wir uns entscheiden, auf welche der vier Spuren wir aufnehmen wollen. Da noch alle unbenutzt sind, wählen wir die erste. Spuren oder auch Tracks nennt man die verschiedenen Speicher, in denen man einen Sound ablegen kann. Man kann übrigens alle vier Spuren auch gleichzeitig abspielen. Aber soweit sind wir noch gar nicht. Wir müssen zunächst die erste Spur anwählen.

Auf der linken Seite des Fensters stehen untereinander die Bezeichnungen Track, Status, Mode... Jeweils auf gleicher Höhe ist dann viermal das gleiche zu finden: Einmal für jede Spur. In der ersten Reihe, die den Namen Track trägt, kann man links ein Kästchen sehen, in dem die Ziffer Eins steht. Ist es rot unterlegt, ist die Spur aktiviert. Wenn nicht, klicken Sie es bitte an. Wo wir gerade dabei sind, können wir auch noch die restlichen Reihen beschrei-

ben. Da wäre zunächst Status. Hier findet man wieder ein Feld wie oben bei Track. Hätten wir auf der Spur bereits etwas aufgenommen, könnten wir durch einen Klick auf dieses Feld dessen Zustand umkehren. Man kann hier einstellen, ob der Sound der zugehörigen Spur gespielt werden soll oder nicht. Schaltet man alle Spuren ein (auf On), hört man beim Abspielen auch mehrere Stimmen durcheinander. Nächster Punkt der Tagesordnung ist Mode. Wie auch bei Status können Sie den Zustand dieses Flags durch einen Mausklick ändern. LOOP und ONCE werden angeboten. Ist LOOP aktiv, wird der Sound, wenn er einmal gestartet ist, immer wieder wiederholt. ONCE sorgt dafür, daß er immer dann, wenn er sein Ende erreicht hat, neu gestartet werden muß.

Alle nun folgenden Reihen sind im Aufbau gleich. In einem länglichen, flachen Kasten steht bei ihnen eine Zahl oder ein Wort. Dahinter kommt ein weiteres Kästchen. Es ist ziemlich quadratisch. Klickt man es an, wird es gefüllt. Fachmännischer ausgedrückt: Es wird aktiviert. Weiterhin wird der Wert, der links von ihm steht, im rechts gelegenen Schieberegler dargestellt. Verschiebt man diesen, kann man den Wert komfortabel ändern. Dazu müssen Sie den Regler anklicken und mit drückter Maustaste innerhalb seines Bereichs verschieben. Damit ändert sich der zugehörige Wert. Da der Track zur Zeit noch leer ist, können Sie das noch nicht ausprobieren. Das Programm würde dann nämlich in einem rot eingerahmten Warnfeld darauf hinweisen, daß ihm dies nicht paßt. Die aufdringliche Meldung verschwindet erst dann wieder, wenn Sie "Continue" anklicken. Mit den zwei Pfeilen, die in den Zeilen Track, Status und Start zu sehen sind, können Sie den Wert, der im Schieberegler dargestellt wird, noch genauer einstellen.

Mit Start und End können Sie den Bereich begrenzen, der gespielt und abgespeichert werden soll. Wenn wir später die Kopierfunktion behandeln, spricht diese auch nur den durch Start und End eingeschlossenen Bereich an. Period und Rate geben an, mit welcher Geschwindigkeit aufgenommen und widergegeben wird. Je höher Period ist, desto langsamer wird aufgenommen und abgespielt. Mit einem hohen Period-Wert kann man folglich mehr aufnehmen als mit einem niedrigen. Das schlägt sich allerdings auch auf die Tonqualität nieder. Rate ändert sich umgekehrt proportional zu Period. Wir werden Rate aber nicht weiter beachten. Volume ist zuständig für die Lautstärke, in der ein aufgenommenes Stück wiedergegeben wird. Interessante Effekte kann man erzielen, wenn man diesen Wert während der Wiedergabe ändert. Die letzte Reihe enthält den sog. OneShot-Wert. Hat man Mode auf LOOP geschaltet, wird das Stück beliebig oft wiederholt. So hatten wir es je-

denfalls formuliert. Zutreffend ist das allerdings nur, wenn OneShot gleich Start ist. Beim ersten Abspielen wird der Bereich von Start bis End wiedergegeben. Bei jedem weiteren Durchgang wird nicht bei Start, sondern bei OneShot begonnen. In IFF-Dateien wird OneShot übrigens gesichert.

Nun, wo Sie alle Einstellungen kennen, die man für jede Spur getrennt vornehmen kann, wollen wir die erste Spur nutzbar machen. Die Eins haben wir schon angeklickt. Jetzt müssen wir uns nur noch Speicher besorgen. Dazu rufen wir die Menüfunktion "Allocate track" aus dem Track-Pulldown auf. Über einen Requester werden wir gefragt, wieviel Speicher wir denn opfern möchten. Für den Anfang genügen uns 100000 Bytes. Klicken Sie die vorgeschlagene Zahl an, und ändern Sie sie in 100000 um. Nun drücken Sie [Return]. An dieser Stelle müssen wir leider auf ein kleines Manko des Programms hinweisen: Es kann Sounds, die zur Zeit nicht benötigt werden, nicht in das Fast-RAM umleiten. Musik kann zwar nur gespielt werden, wenn Sie im Chip-RAM liegt, aber die Teile, die der Soundchip gerade nicht benötigt, könnte man ruhig umlagern. Entweder kann man dazu das Fast-RAM oder eine Diskette verwenden. Dadurch hätte man die Möglichkeit, längere Stücke abspielen und aufnehmen zu können. Wenn Sie unbedingt längere Stücke verwenden wollen, können wir Sie auf den Deluxe Soundsampler mit der dazugehörigen Software verweisen. Mit diesem recht teuren Paket können Sie direkt auf Diskette aufnehmen und so bis zu 880 KByte lange Digitalsounds speichern.

Doch weiter. Der Track hat einen Speicherblock zugewiesen bekommen. Nun stellen wir den Period-Wert so ein, wie wir ihn haben möchten. Doch welcher Wert ist der richtige? Das richtet sich ganz nach den Ansprüchen. Mit einem Wert von 550 kann man man lange und einigermaßen gut aufnehmen. Legt man Wert auf besondere Qualität, fährt man mit 200 ganz gut. Später können Sie ja ausprobieren, welchen Wert Sie bevorzugen.

Für eine verwendbare Aufzeichnung muß am Eingang des Samplers eine vernünftige Lautstärke anliegen. Beim Alcomp-Digitalisierer muß man die Lautstärke (und damit Stromstärke) verwenden, mit der normalerweise ein kleiner Lautsprecher betrieben wird.

Nachdem Sie eine Tonquelle angeschlossen haben, können Sie mit der Aufnahme beginnen. Dazu klicken wir das Feld "Mouse Rec" an. Es befindet sich mit einigen anderen Gadgets (so nennt man die anklickbaren Felder) ganz unten im Fenster. Wenn Sie nun der Aufforderung folgen, die Maustaste er-

neut zu drücken, beginnt die Aufzeichnung. Überprüfen Sie, ob das Gerät, von dem Sie aufnehmen wollen, startbereit ist. Dann drücken Sie die linke Maustaste und Starten den Kassettenrekorder, CD-Player oder was Sie sonst verwenden. Freundlich, wie er ist, informiert Sie der Amiga nun, daß er mit der Aufzeichnung begonnen hat. "Recording..." nennt er das. Verschwindet die Anzeige, ist der Speicher der Spur voll.

Hören wir uns doch einmal an, was wir aufgezeichnet haben. Es ist ratsam, erst einmal für Ruhe im Raum zu sorgen. Dann klicken wir auf PLAY. Wenn alles geklappt hat, hören wir nun das, was wir aufgezeichnet haben.

Sollten Sie nichts hören, kann das an zwei Dingen liegen: Entweder Status ist auf OFF geschaltet (in diesem Fall ändern Sie dies) oder die Aufnahme hat nicht geklappt. Fehlerquellen gibt es bestimmt viele. Wir haben einige Fehler (wenn man das überhaupt so nennen darf) gemacht und daraus gelernt. Diese Fehler möchten wir Ihnen vorstellen und zeigen, wie man mit Ihnen fertig wird.

Sie haben PLAY aufgerufen und hören nichts. Alles halb so wild: Sie haben ja uns. Häufigste Fehlerquelle ist das Status-Feld. Wir sagten das schon. Überprüfen Sie, ob die Musik eingeschaltet ist (ON). Nächster Punkt der Tagesordung ist Volume. Überprüfen Sie, ob die Lautstärke nicht zu niedrig eingestellt ist. Rufen Sie nach einer Änderung zur Sicherheit nochmal "PLAY" auf. Wenn sich immer noch nichts tut, warten Sie bitte ein bißchen. Es kann sein, daß Sie mit der Aufnahme begonnen haben, während die Tonquelle noch kein Signal lieferte. Ursache dafür kann z.B. sein, daß Sie Ihr Radio oder Ihren Rekorder etwas später gestartet haben als die Aufnahme am Amiga. Verwenden Sie einen Plattenspieler, müssen Sie einen weiteren Toleranzbereich einräumen. Bevor dieser nämlich die ersten Töne von sich gibt, muß er einige Umdrehungen auf den Leerspuren am Rand der Platte machen. Unter Umständen muß sein Tonkopf auch noch auf die Platte bewegt werden und und und... Bei all diesen Verzögerungen kann es schon mal vorkommen, daß der Amiga die Aufnahme beendet, bevor die ersten Töne eintreffen. Besonders dann, wenn man wenig Speicher bereitgestellt hat und mit einer hohen Geschwindigkeit aufnimmt, ist das der Fall. Versuchen Sie es ruhig mehrmals.

Es kann auch nichts schaden, die Anleitung des Digitalisierers genau zu studieren. Vielleicht enthält sie einen Hinweis, der speziell für Ihren Sampler wichtig ist. Zum Beispiel könnte dort stehen, daß Sie ihn erst justieren müssen...

Zunächst gibt es aber noch einige allgemeine Fehlerquellen. Da wäre z.B. der Sampler selbst zu nennen. Sicher haben Sie ihn angeschlossen. Oder? Aber haben Sie auch den zweiten Stecker dort eingesteckt, wo es die Anleitung angibt? Nicht? Dann wird es höchste Zeit. Der Alcomp-Sound-Sapler funktioniert z.B. nur, wenn er am zweiten Mausport angeschlossen wird.

Mit Steckern geht es bei der Fehlersuche weiter. Haben Sie die Leitung, die den Sampler mit der Tonquelle verbindet, richtig eingesteckt? Unser Digitalisierer erhält das, was er aufnehmen soll über eine Cinch-Buchse. Steckt man den Stecker zu zaghaft ein, weil man glaubt, etwas beschädigen zu können, kann es sein, daß der äußere Kontakt nicht verbunden wird. Erkennbar ist dies oft an einem Brummen. Der Stecker muß so weit eingeschoben werden, daß der Ring, der dessen Pin umgibt, fest auf der Metallummantelung der Kupplung sitzt. Überprüfen Sie das bitte.

Sind Sie ganz sicher, daß Sie das richtige Kabel verwenden? Ja? Gut! Aber haben Sie auch den richtigen Cinch-Stecker eingesteckt? Über die Methode des Ausprobierens können Sie das überprüfen. Unser Sampler muß mit einem Lautsprecher- oder Kopfhöreranschluß verbunden werden. Haben Sie etwas aufgenommen, was viel zu leise ist, kann es sein, daß Sie aus Versehen eine Verbindung zu einem Geräteanschluß hergestellt haben. Diese führen nämlich eine viel niedrigere Spannung und sind daher auch viel leiser.

Haben Sie die Lautstärke Ihrer Tonquelle richtig eingestellt? Sehen Sie doch einmal nach. Eine bis jetzt ganz außer Acht gelassene Fehlerquelle ist die Verbindung des Amiga mit dem Monitor. Sind beide(!) Audiokanäle angeschlossen? Auch dann, wenn Sie einen Monitor oder Fernseher ohne Stereo verwenden, müssen Sie dies beachten. Haben Sie die Lautstärke des Monitors überhaupt aufgedreht? Diese Frage ist gar nicht so dumm, wie sie sich anhört. Einige Programme erzeugen nämlich viel zu laute und entsetzlich schiefe Töne. Wen wunderts, wenn man dann den Ton einfach abstellt?

Eine letzte Fehlerquelle ist der 1081/1084-Monitor. Er erhält sowohl sein Bild, als auch seinen Ton über eine Euro-AV-Buchse. Da Commodore ein entsetzlich dickes Kabel mit dem Monitor ausliefert, kann es leicht sein, daß sich der Scart-Stecker etwas lockert. Scart? Ja genau. Ob man Scart oder Euro-AV sagt, ist egal. Lockert sich dieser Stecker, haben einige Pins keinen Kontakt mehr. Dies kann sich darin äußern, daß eine Farbe ausfällt, gar nichts mehr zu sehen ist oder aber daß einer oder beide Tonkanäle ausfallen.

Wenn Sie alle oben genannten möglichen Fehler durchchecken, müßte eigentlich alles klar gehen. Wir können uns nun einigen weiteren Funktionen zuwenden. Im Track-Menü finden wird außer "Allocate track" noch einige andere interessante Dinge. "Mix track" und "Display track" hören sich z.B. recht vielversprechend an. Bevor wir uns allerdings über die tollen Effekte freuen, die wir damit erzielen können, sollten wir uns noch zwei Funktionen ansehen, die zwar nicht gerade interessant, dafür aber wichtig sind. "Clear track" und "Clear all tracks" sind die beiden Menüpunkte, um die es hier geht. Der erste gibt den Speicher des gewählten Tracks (also der Spur) frei, der zweite tut das gleiche mit allen Spuren.

"Copy track" dupliziert den Bereich von "Start" bis "End" der gewählten Spur. Nachdem Sie diese Funktion aufgerufen haben, werden Sie gefragt, wohin denn kopiert werden soll. Sie müssen die Nummer der Spur anwählen.

"Concat track" funktioniert ähnlich. Hierbei wird der Bereich jedoch an die Spur angehängt, die man als Ziel (Destination) angibt. Um die zusammengeschnittene Spur zu hören, muß man allerdings den "End"-Wert der Spur hochstellen.

"Mix track" dient dazu, zwei Spuren zu einer einzigen zusammenzufassen. Dabei wird die Spur, die beim Aufruf der Funktion aktiv ist, mit einer anderen verbunden, deren Nummer man genauso wie bei Copy und Concat einstellen muß. Das Ergebnis dieser Verknüpfung wird auch in dieser Spur abgelegt.

"Reverse track" dreht den Bereich einer Spur um. Aus Gartenschlauch wird hcualhcsnetraG. Die noch verbleibende Funktion "Diskplay track" zeigt den Tonverlauf der Spur an. Man kann sich verschiedene Bereiche ansehen. So toll und professionell das, was diese Funktion ausgibt, auch aussieht, anfangen kann damit meist nicht besonders viel.

Sind Sie auf den Geschmack gekommen, was man mit dem Sampler so alles machen kann? Wenn Sie sich einige Public-Domain-Disketten bestellen, auf denen sich Hilfsmittel zum Abspielen und Weiterverarbeiten von digitalisierter Musik befinden, können Sie sogar Disketten anlegen, die nach einem Reset die von Ihnen aufgenommene Musik oder Sprache im Hintergrund abspielen. Es wäre sogar ohne weiteres möglich, eine Information über ein Mikro aufzunehmen und auf Ihrer Diskette abzulegen. So hören Sie (und andere) immer Ihre Stimme aus dem Computer.

Auch nicht verachtet werden sollte die Möglichkeit, die Sounds gewerblich zu nutzen. So können Sie z.B. einen Versandhandel mit digitalisierten Sounds betreiben - oder in Verbindung mit einem Musikprogramm, einer MIDI-Schnittstelle und anderen Raffinessen sogar ein professionelles Lied produzieren. Was Sie auch vorhaben - wir wünschen Ihnen viel Erfolg. Sollten Sie eine Single herausbringen, schreiben Sie uns doch mal.

4.5 Datenaustausch mit anderen Computern

Kommunikation ist das Wort der Zukunft. Über Telefon und Briefe unterhalten sich die Menschen - Über Telefon, Datex-P, Telex, BTX, Telefax und viele andere Medien reden die Computer ein Wörtchen mit. Auch der Amiga kann mit all diesen Medien Kontakt aufnehmen. In der Bundesrepublik ist das zur Zeit jedoch noch so eine Sache. Die Post erlaubt es nicht, postfremde Geräte anzuschließen. Eine Ausnahme bestätigt die Regel: Geräte mit einer sogenannten FTZ-Nummer. Bewegt man sich im Rahmen der Legalität, ist man stark eingeschränkt, da von der Post zugelassene Modems nicht nur teuer, sondern obendrein meist noch schlecht sind. In absehbarer Zeit wird man von dieser Regelung aber zurücktreten, da sie ohnehin umgangen wird.

Ein Modem ist übrigens ein Gerät, das auf der einen Seite an das Telefonnetz und auf der anderen an einen Computer angeschlossen wird. Mit einem solchen Gerät kann man Daten zwischen verschiedenen Geräten austauschen. Auch kann man einige Großrechner über das Datex-P-Next erreichen. Dazu muß man sich allerdings eine Datex-P-Teilnehmerkennung geben lassen. Über eine Vermittlungsstelle kann man dann die verschiedenen Firmen und Geräte erreichen. Über dieses System arbeiten auch viele Hacker. Hacker sind Computeranwender, die mit Erfolg versuchen, Paß- bzw. Codewörter für diverse Rechner ausfindig zu machen und über diese in fremde Rechnersysteme einzudringen. Ein durchaus interessantes, aber strafbares Hobby. Teuer ist es auch noch. Dabei sind in diesem Fall nicht die Telefonkosten (die bezahlen erfolgreiche Hacker nämlich nicht), sondern Unkosten gemeint, die bei der Suche nach Paßwörtern entstehen. So reisen Hacker z.B. zu Computermessen, wo sie mit einer Kamera bewaffnet darauf warten, daß ein gestreßter Firmenvertreter in sein System einloggt. Einloggen nennt man den Vorgang der Kontaktaufnahme. Bei der Eingabe des Paßwortes, weicht das Teleobjektiv nicht von seinen Fingern ab. Tja - und dann hat unser Hacker wieder ein neues Wort in seiner Liste, das er seinen Freunden weitergibt. Eine andere Methode,

an Paßwörter zu kommen, ist das einfache ausprobieren. Viele Rechner sind nämlich nicht über so unknackbare Codes wie KVM_OEL6M&, sondern über normale Worte zu erreichen. Oft kann man Paßwörter verwenden, die durch Einfallslosigkeit oder Schlampigkeit noch im System sind. So werden z.B. alle VAX-Rechner mit einem Standard-Paßwort ausgeliefert. Wenn die Firma, die den Rechner installiert, dieses Paßwort nicht entfernt, haben Hacker ein leichtes Spiel. Sie testen einfach aus, ob das Paßwort noch gespeichert ist. Wenn ja, haben Sie vollen Zugriff. Bei einer VAX muß man als Benutzername "system" und als Paßwort "manager" verwenden - schon ist man drin! Es gibt noch unzählige andere Verfahren, mit denen man Paßwörter illegal erfahren kann. Neben den verrücktesten Tricks gibt es noch einen, der so einfach wie genial ist. Man hört das Telefon ab. Dazu muß man nicht einmal die Leitung anzapfen. Es reicht aus, ein Tonbandgerät mit einem induktiven Mikrophon in die Nähe des Modems oder der Telefonleitung zu legen, über die ein Computerbenutzer sein Paßwort schickt. Das aufgezeichnete Geräusch kann mit einem Akkustik-Koppler in Klartext umgewandelt werden - und schon hat man sein Paßwort. All diese Dinge sind (verständlicherweise) strafbar. Viele ehemalige Hacker sind aber nicht dort gelandet, wo man Sie vermuten sollte, nämlich im Knast, sondern in Rechenzentren und Computerfirmen, wo sie für die Sicherheit sorgen!

Doch zurück zur Legalität: Eine Alternative zum Modem ist der Akkustikkopler. Er ist zwar nicht so professionell, aber für den Privatbereich ausreichend. Ein gutes, aber nicht ganz billiges Gerät ist das Dataphon S21/23D. Es kostet ca. 500 DM, ist BTX-fähig, kann mit zwei Geschwindigkeiten arbeiten und wird im Induktiv-Modus nicht von Nebengeräuschen gestört. Wenn Sie beabsichtigen, mit einem Koppler Ferngespräche zu Mailboxen zu führen, sollten Sie eines wissen: Eine monatliche Telefon-Rechnung von einigen hundert oder sogar tausend Mark rückt in den Bereich des leicht erreichbaren. Ich greife dies nicht etwa aus der Luft - ich spreche aus Erfahrung!!!! Um Daten innerhalb des Ortsnetzes zu übertragen, ist ein Koppler jedoch das ideale Gerät. Schon für 250 DM kann man ein gutes Gerät erwerben.

Oben fiel der Begriff Mailbox. Dazu sind wir Ihnen wohl eine kleine Erklärung schuldig. Viele Firmen, Institute, Vereine und Computer-Clubs betreiben Mailboxen. Aus diesen Mailboxen kann man Daten und Programme herausholen. Man kann auch Programme, Texte und Briefe dort ablegen. Leider sind gute Mailboxen nicht immer in der Nähe. Das billige Ortstarifvergnügen ist also nicht für alle zu haben.

Bei all dem Für und Wieder noch eine Kleinigkeit zur Entscheidung Koppler oder Modem. Unser Rat: Verwenden Sie einen Koppler. Modems sind teurer. Auch von solchen, die in der Bundesrepublik nicht betrieben werden dürfen, müssen wir Ihnen abraten. Man kann Sie zwar theoretisch anschließen. Dies ist aber zur Zeit noch verboten. Und da versteht die Post überhaupt keinen Spaß. Mit technischen Tricks kann sie prüfen, ob an einem Anschluß ein weiteres oder anderes Gerät angeschlossen wurde. Also Finger weg!

4.5.1 Nullmodems

Ähnlich wie Koppler und Modems funktionieren Nullmodems. Sie unterscheiden sich jedoch in zwei Dingen: Erstens wandeln Sie die Daten nicht in Töne um, wie das von Kopplern und Modems getan wird, und zum anderen sind sie viel billiger. Sie kosten im Eigenbau ca. 15 DM. Toll! Aber wo ist der Haken? Nun ja. Mit Ihnen überträgt man nur in reinen Datennetzen. Also direkt von Computer zu Computer.

Über ein Nullmodem (und eventuell noch ein RS232-Verlängerungskabel) kann man Daten zwischen verschiedenen Rechnern austauschen. Bei zwei Amiga ist das meist sinnlos. Aber... Folgendes Beispiel: Angenommen, am Arbeits- oder Ausbildungsplatz verwenden Sie PC. Ein PC ist ein verbeiteter Computer, der zwar nicht ganz auf dem neuesten Stand der Technik, also auf dem des Amiga, ist, aber auch Vorteile bietet. In Kapitel 7 können Sie noch näheres darüber erfahren. Aber wieder zu unserem Beispiel. Sie erhalten den Auftrag, einen Text auf dem PC zu schreiben. Dazu müssen Sie, wenn Sie keinen eigenen PC oder eine PC-Simulation auf dem Amiga haben, immer dann, wenn Sie daran arbeiten wollen, an den Arbeitsplatz bzw. zu Ihrer Ausbildungsstelle gehen. Schöner wäre es doch, den Text zu Hause am vertrauten Amiga zu schreiben und ihn dann, wenn er fertiggestellt ist, auf den PC zu übertragen. Dazu können Sie ein Nullmodem verwenden. Zwar sind auch Koppler verwendbar, aber sinnvoll sind sie hier nicht.

Bevor Sie nun, begeistert durch unseren Vorschlag, beginnen, etwas ähnliches zu tun, sollten Sie zunächst zwei Dinge klären: Erstens müssen Sie entweder den Amiga zum PC oder den PC kurz zum Amiga bringen. Sollte ein Bekannter von Ihnen sowohl einen PC als auch einen Amiga besitzen, können Sie ihm auch ein paar kleine Geschenke machen und ihn ganz nebenbei fragen, ob Sie nicht mal kurz die Geräte benutzen könnten.

Sie müssen dies insgesamt zweimal tun. Zuerst müssen Sie testen, ob sich das, was Sie übertragen wollen, auch ohne größere Probleme übertragen läßt. Geht bei diesem Test alles glatt, können Sie ihr Projekt in Angriff nehmen und es später zum PC schicken.

Besitzen Sie oder ein Bekannter von Ihnen eine PC-Karte für den Amiga oder ein SideCar, brauchen Sie sich diese Mühe gar nicht zu machen. Im Besitz eines dieser Zusatzgeräte können Sie eine Datei vom PC zum Amiga übertragen und umgekehrt. Auch kann man mit dem Programm DOS-2-DOS Disketten auf dem Amiga im PC-Format beschreiben. Dazu muß der PC allerdings ein 3½-Zoll- oder der Amiga ein zusätzliches 5¼-Zoll-Laufwerk haben. Doch Vorsicht: Viele externe Fremdlaufwerke, die unter AmigaDOS einwandfrei funktionieren, sind nicht in der Lage, PC-Disketten richtig zu bearbeiten. Hier empfiehlt sich vor dem Kauf beim Händler ein kleiner Test mit DOS-2-DOS, einer PC-Karte oder der MS-DOS-Software-Emulation. Beschränken Sie sich bei diesem Test jedoch auf keinen Fall auf das Lesen des Inhaltsverzeichnisses! Dieses liegt auf PC-Disketten nämlich auf Spur Null. Diese Spur kann auch mit nicht-PC-tauglichen Geräten meist einwandfrei gelesen werden. Um die Funktionsfähigkeit auf allen Spuren zu testen, sollte man eine normale Diskette mit dem Diskcopy-Befehl von MS-DOS im externen Laufwerk kopieren und diese dann auf einem richtigen PC testen. Wenn dieser Test auch mit vertauschten Rollen keine Schwierigkeiten macht, spricht eigentlich nichts gegen das Laufwerk. Ansonsten: Finger weg!

4.5.2 Anschluß und Anwendung

Ob Koppler, Modem oder Nullmodem - alle müssen über Software gesteuert werden. Dazu kann man entweder ein spezielles DFÜ-Programm oder diverse Funktionen aus verschiedenen Programmen verwenden. Beides hat vor und Nachteile. DFÜ-Programme sind zwar vielfältig, aber nicht auf spezielle Probleme zugeschnitten. Die Funktionen der Anwendungsprogramme sind speziell auf die zu übertragenden Daten ausgerichtet und daher, wenn möglich zu bevorzugen.

Allgemein verwendet man jedoch ein DFÜ- oder auch Terminal-Programm. Ein solches Programm zeigt an, was über die Datenleitung hereinkommt. Über die Tastatur kann man auch Meldungen nach draußen schicken. Es würde den Rahmen des Möglichen sprengen, hier alles zu erzählen, was es zu

DFÜ zu sagen gibt. Wir werden uns daher darauf beschränken, darzulegen, wie die Geräte angeschlossen werden und wie man sie mit den richtigen Parametern betreibt. Ohne die läuft nämlich nichts.

Als Beispiel verwenden wir das auf der ExtrasD abgelegte Programm mit dem Namen Terminal. Es ist ein einfaches DFÜ-Programm, an dem sich die wichtigsten Funktionen darstellen lassen.

Bevor wir nun dieses oder ein anderes Programm starten, müssen wir zunächst den Koppler, das Modem oder das Nullmodem anschließen. Dazu verwenden wir die serielle Schnittstelle. Sie liegt zwischen dem parallelen Port, an dem Drucker und Digitalisierer betrieben werden, und dem Anschluß für externe Laufwerke. Beim Amiga 500 kann hier ein ganz normales RS232-Kabel angeschlossen werden. RS232 ist eine andere Bezeichnung für diesen Port.

An dieser Stelle sei uns noch ein kleiner Hinweis gestattet. Nullmodems sind schwer zu bekommen, da sie nicht oft gekauft werden. Und der Anwendungskreis, der sie verwendet, ist meistens auch in der Lage, sie selbst herzustellen. Sollten Sie nicht zu den Lötkolbenfans gehören, sollten Sie versuchen, ein Nullmodem zu kaufen. Glück könnten Sie bei einem der vielen Elektronikversender haben. So haben wir z.B. in einem Katalog der "Westfalia Technica" ein Nullmodem gefunden. Es kostete dort 12,95 DM und muß noch mit einem Kabel verbunden werden. In Kapitel 4.6.6 finden Sie übrigens eine Anleitung zum Bau eines Nullmodems.

Aber kommen wir endlich zur Anwendung. Die RS232-Schnittstelle kann über verschiedene Parameter betrieben werden. Alle Mailboxen geben z.B. an, über welche Parameter sie angesprochen werden können. Sie müssen die vorgegebenen Werte dann in Ihrem DFÜ-Programm einstellen, um mit der Box Kontakt aufnehmen zu können. Wenn das DFÜ- oder Anwendungsprogramm keine Einstellmöglichkeiten bietet, können Sie die Parameter meist über Preferences einstellen. Mit "Change Serial" kommen Sie in das entsprechende Untermenü. Wenn Sie zwei Computer über ein Nullmodem verbinden, müssen beide mit den gleichen Parametern arbeiten. Je nach Art der Daten muß man noch verschiedene Vorgaben beachten. Doch dazu später mehr.

Allgemein üblich ist eine dreistellige Parameterangabe, die eventuell noch um die Übertragungsgeschwindigkeit ergänzt wird. Geben wir diesen Parametern

zur Veranschaulichung die Namen X, Y und Z. Man kann diese auf zwei Arten notieren: Entweder XYZ oder X/Y/Z. Soso! Und was heißt das nun?

Beginnen wir mit X. Für X kann man eine Zahl einsetzen. Diese Zahl gibt an, wie groß die Werte sind, die man übertragen möchte. Dabei wird die Anzahl der Bits angegeben. Möchte man also ganze Bytes (größer geht es nicht) übermitteln, muß man hier eine acht angeben. Oft reichen aber auch sieben Bits. Leider verwenden die meisten Mailboxen auch nur sieben.

Weiter geht es mit Y. Hier kann man festlegen, ob und - wenn ja - wie, eine Prüfsumme gebildet werden soll. Dafür gibt man einen der Buchstaben N, O oder E an. Bei N wird keine Prüfsumme gebildet. O steht für "odd" und E für "even". Diese englischen Begriffe können mit ungerade und gerade übersetzt werden. Durch die Addition aller Bits (einschließlich eines Prüfbits) muß entwender ein gerader oder ungerader Wert herauskommen - je nach der gewählten Prüfmethode.

Unsere dritte Unbekannte ist Z. Für Z kann man "1" oder "2" einsetzen. Diese Zahl gibt die Anzahl der Stoppbits an, mit der die Daten übertragen werden sollen. Nehmen Sie das bitte einfach so, wie es ist. Warum man Stoppbits verwendet, muß nur die Konstrukteure der Geräte und die Programmierer interessieren, die die System-Software zum Betrieb der Schnittstelle schreiben.

Nun haben wir alle Parameter behandelt. Fehlt nur noch eine Kleinigkeit: Die Übertragungsgeschwindigkeit. Sie wird in Baud angegeben. Bei einer Übertragungsgeschwindigkeit von 300 Baud werden 300 Bits pro Sekunde übermittelt. Das sind ca. 37 Bytes. Mit 300 Baud arbeiten die meisten Mailboxen. Verbreitet sind aber auch 1200 Baud. Wenn Sie zwei Amiga direkt miteinander verbinden, können Sie auch 9600 Baud verwenden. Damit kann man sogar große Programme und Texte schnell übergeben. Koppler arbeiten meistens mit 300, maximal mit 1200 Baud.

In unserem Terminal-Programm von der ExtrasD-Diskette können wir über ein Menü sowohl die Baudrate, also die Übertragungsgeschwindigkeit, als auch einige Parameter einstellen. In der Kopfleiste des Fensters können wir lesen, daß das Modem noch nicht bereit ist. Die Ursache dafür ist nicht etwa, daß wir etwas falsch gemacht haben. Vielmehr muß man die Kommunikation erst freigeben. Dazu rufen wir im ersten Menü "On Line" auf. Zur Bestätigung wird nun MODEM BEREIT angezeigt. Das Programm geht davon aus, daß

Sie mit einer Mailbox in Verbindung treten wollen. Eine Mailbox schickt nämlich die Zeichen, die Sie eingeben, nochmals zurück. So können Sie mögliche Übertragungsfehler erkennen. Wenn Sie mit einem System Kontakt aufnehmen wollen, das keine Zeichen zurückschickt, z.B. ein anderer Computer, dem Sie einen Text schicken wollen, können Sie durch Anwahl von "Echo" erreichen, daß alle Zeichen, die Sie eingeben, auch angezeigt werden.

Mit unserem Terminalprogramm können wir auch Dateien übertragen und empfangen. Dazu stehen zwei Menüfunktionen bereit. Beachten Sie aber, daß viele Sonderzeichen nur dann richtig übermittelt werden können, wenn Sie den 8-Bit-Datenaustausch aktiviert haben. Weiterhin kann es sein, daß das Gerät, dem Sie Ihren Text oder Ihr Programm übermitteln, bestimmte Zeichen nicht kennt oder sie mit anderen verwechselt. Im ersten Fall müssen Sie wohl oder übel ein anderes Zeichen verwenden. Im anderen Fall können Sie sich jedoch mit einem kleinen Trick behelfen: Bevor Sie mit der Übertragung beginnen, tauschen Sie alle Sonderzeichen mit einer Textverarbeitung und geben Zeichenketten aus, die Sie bestimmt an keiner anderen Stelle verwendet haben. Also:

```
Tausche Ä gegen &&AE&&
Tausche ä gegen &&ae&&
Tausche Sonderzeichen xx gegen &&S01&&
```

Dann übermitteln Sie die so geänderte Datei. Auf dem anderen Computer können Sie die Zeichen wieder richten:

```
Tausche &&AE&& gegen AE
Tausche &&ae&& gegen ae
Tausche &&S01&& gegen Sonderzeichen xx
```

Wenn Sie BASIC-Programme von einem Rechner zum anderen übertragen, müssen Sie diese vorher mit SAVE "Ascii",A in ein lesbares Format umwandeln. Wichtig ist hier das ",A". Es wandelt das interne Format, das nur vom BASIC-Interpreter verstanden wird, in ein anderes um, das alles in Klartext darstellt. Die so umgewandelte Datei können Sie dann auf ein anderes Gerät übertragen (zum Beispiel auf einen PC mit GW-BASIC). Beachten sollten Sie aber, daß das Programm schon auf der Amigaseite Zeilennummern erhalten hat, da die meisten BASIC-Interpreter diese erwarten.

Wenn wir Ihnen auch keinen vollständigen Einblick in die DFÜ, die Kommunikation mit Mailboxen und all die anderen Dinge geben konnten, hoffen wir doch, Ihnen ein universelles Grundwissen vermittelt zu haben. Wir wünschen "Gut hack!"

4.5.3 Ein externes CLI

An Großrechnern können mehrere Benutzer gleichzeitig arbeiten. Jeder Arbeitsplatz ist mit einem Terminal ausgestattet, über das man auf den eigentlichen Rechner zugreifen kann. Dieses Terminal ist nichts anderes als ein kleiner Computer. Wenn man nun hört, daß der Amiga in Verbindung mit einer Turbo-Karte schneller als einige der bekanntesten Großrechner ist, kommt die berechtigte Frage auf, ob es nicht möglich ist, über verschiedene Computer auf den Amiga zuzugreifen.

Ohne weiteres ist es leider nicht möglich. Man kann, wenn man keine umfangreiche Hard- oder Software installiert, nur mit einem externen Terminal arbeiten. Dabei kann man einen anderen Computer verwenden, den man noch auf Lager hat oder den man eigentlich schon verschrotten wollte. (Im letzten Fall können wir einen Atari ST oder einen IBM WC empfehlen, obwohl es bei diesen Geräten mit den hohen Übertragungsgeschwindigkeiten des Amiga Probleme geben könnte!)

Spaß beiseite - kommen wir wieder zu unserem Terminal: Wir schließen es über einen Nullmodem an die serielle Schnittstelle an. Doch, wie können wir es benutzen? Hier sind wir, wenn wir uns im Rahmen eines vertretbaren Rahmens bewegen, eigentlich auf das CLI eingeschränkt. Es läge auf der Hand, einfach ein CLI mit

```
NewCLI SER:
```

auszulagern. Wie gesagt: es läge, wenn... Ja, wenn der Amiga-Treiber der seriellen Schnittstelle für diesen Fall ausgelegt wäre. Richtig kann man leider nur mit Workbench 1.3 arbeiten. Dort kann man mit

```
Mount Aux:
NewCLI Aux:
```

ein CLI auf einem anderen Rechner verwenden, auf dem natürlich ein geeignetes Terminal-Programm laufen muß.

4.6 Kleine Basteleien am kleinen Amiga

Auch den besten Computer kann man noch verbessern - keine Frage. Aber wußten Sie schon, daß man den Amiga 500 preiswert umbauen kann, ohne ein Elektronikstudium absolviert zu haben? Egal, wie Sie diese Frage beantworten - in diesem Kapitel finden Sie bestimmt eine Anregung, den Amiga oder dessen Monitor umzubauen. Wir zeigen Ihnen z.B., wie Sie einen Amigamonitor in Verbindung mit einem Video-Rekorder oder einem TV-Tuner zu einem vollwertigen Fernseher umrüsten, dessen Bild exzellent ist. Aber nicht nur das. Der Monitor-Fernseher spart auch viel Geld! Weiterhin stellen wir Ihnen einen neuen Prozessor vor, der etwas schneller als der 68000er arbeitet. Doch das sind nur zwei von vielen Möglichkeiten. Im weiteren Verlauf des Kapitels werden wir Ihnen noch einige andere vorstellen. Lassen Sie sich also überraschen, welche Tricks wir noch auf Lager haben...

4.6.1 Kickstart-Umschaltung

Als der Amiga 500 auf den Markt kam, wurde er oft und viel kritisiert, weil er sein Kickstart nicht mehr von Diskette lädt. Es gab zu dieser Zeit nämlich noch viele Programme, die nur mit Kickstart 1.1 liefen und daher nur auf einem Amiga 1000 verwendet werden konnten. Beim alten 1000er konnte bzw. mußte man nämlich das Betriebssystem, also das Kickstart, von Diskette laden. Da der 500er mit der Kickstart-Version 1.2 ausgeliefert wurde, funktionierten einige Programme nicht. An diesem Zustand hat sich aber mittlerweile einiges geändert. Zum einen haben die Software-Hersteller ihre Produkte angepaßt, und zum anderen kann man heute mittels einer Kickstart-Umschaltung selbst wählen, mit welcher Version des Bestriebssystems man arbeiten möchte. Solche Umschaltungen werden von verschiedenen Firmen, z.B. Alcomp und Message, angeboten.

Mittlerweile redet fast niemand mehr vom Problem der Inkompatibilität zwischen Software für Kick 1.1 und Kick 1.2. Zur Zeit sind ganz andere Versionen im Gespräch. Heiß begehrt ist z.B. Kickstart 1.3.

Um die Kickstart-Umschaltung zu installieren, müssen wir den Amiga wieder einmal öffnen. Rechts neben dem 68000er finden wir das ROM, d.h. den Speicher, der beim ausschalten nicht verloren geht. In ihm befindet sich das Kickstart. Hebeln Sie diesen Chip mit seinen 40 Pins bitte vorsichtig aus.

Dann setzen Sie die Umschaltplatine in dessen Sockel ein.

Das ROM wird auch wieder eingebaut. Auf der Umschaltung befindet sich ein dafür vorgesehener Sockel. Nun stellt sich die Frage, woher man die Eproms, d.h. die Speicher-Chips, in denen sich das neue Kickstart befindet, bekommen soll. Die Firma Message bietet deshalb einen Eprombrennservice an.

Zwei Geräte im Vergleich

Alcomp:

Die Kickstart-Umschaltung von Alcomp gestattet es, zwei neue Kickstart-Sätze zu je vier Eproms (Speicherbausteine) einzusetzen. Allerdings ist die Platine dadurch auch ganz schön groß. In unserem Amiga 500 mußten wir eine (relativ unwichtige) Blechstütze umbiegen, um sie einsetzen zu können.

Message:

Der Umschaltsatz von Message kann nur mit einem zusätzlichen Epromsatz bestückt werden. Sie ist, da sie kleiner ist, leichter einzubauen. Auch kann die Blechstütze in ihrer Position bleiben.

4.6.2 Der preiswerte Fernseher

Wir sagten es bereits in der Einleitung: Man kann den Monitor des Amiga leicht zum Fernseher umrüsten. Dazu muß man entweder einen Video-Rekorder oder einen TV-Tuner besitzen.

In beiden Fällen geht man ähnlich vor.

Über den Video-Rekorder

Verwenden Sie den Commodore-Monitor 1081 bzw. 1084, können Sie ihn leicht als Fernseher betreiben. Natürlich ist er auch weiterhin für den Amiga verfügbar. Da er über mehrere Signaleingänge verfügt, können beide Bilder abwechselnd verwendet werden, ohne daß ein Stecker umgesteckt werden muß. Weiterhin ist es sogar möglich ein drittes Gerät anzuschließen, das dann allerdings ein besonderes Signal bereitstellen muß, das eigentlich nur bei PC verwendet wird. Für alle, die es interessiert: gemeint ist das RGB-Digital-Signal, mit dem man nur 16 Farben darstellen kann. Etwas professioneller ist übrigens die Bezeichnung "TTL-RGB".

Der Amiga benutzt ja bekanntlich auch ein RGB-System. Man nennt es allerdings RGB-Analog. Aus dem Wort kann man schließen, daß ein analoges Signal verwendet wird. Man kann mit dem analogen RGB-System des Amiga viel mehr Farben darstellen, als mit dem digitalen des PC. Doch zurück zu unserem Problem.

Sehen wir uns einmal an, was der 1081 an Schnittstellen zu bieten hat. Um Panik zu vermeiden, sagen wir Ihnen schon jetzt, daß ein anderer Monitortyp auch andere Anschlüsse haben kann. Besitzen Sie z.B. einen Stereo-Monitor, müßte dieser über zwei getrennte Toneingänge verfügen.

Die größte Buchse an der Rückseite des Monitors ist für den Amiga vorgesehen. Man hat hier das Scart- oder auch Euro-AV-Format verwendet, das auch bei Video-Rekordern und Fersehern wegen seiner Einfachheit immer beliebter wird. Sollte Ihnen nun der Gedanke durch den Kopf schießen, daß Sie den Scart-Stecker, über den das RGB-Signal des Amiga ausgegeben wird, an einen Fernseher oder Video-Rekorder anschließen könnten, müssen wir Sie leider in den meisten Fällen enttäuschen. Diese Gerät haben nämlich mit RGB meist nicht viel am Hut - bzw. an den Schnittstellen. Sie verwenden FBAS. Ausprobieren sollten Sie es auf keinen Fall, ohne einen Fachmann um Rat zu fragen. Der Amiga verwendet nämlich höhere Spannungen, als viele Fernseher verkraften können. Mit einem kleinen Trick ist es aber doch möglich, ein TV-Gerät mit Scart-Eingang als Monitor zu betreiben.

FBAS mischt alle drei Farben auf einer einzigen Leitung. Entsprechend schlechter ist auch die Bildqualität. Hat Ihr Rekorder entgegen allen Erwartungen einen RGB-Ausgang, werden Sie sich doch mit dem Video- (= FBAS) Signal begnügen müssen. Der Grund dafür ist denkbar einfach. Der analoge

RGB-Eingang ist ja schon vom Amiga belegt. Da keine zweite Scart-Buchse am Monitor vorhanden ist, begeben wir uns nun auf die Suche nach einem geeigneten Anschluß. Wir werden auf der linken Seite fündig - haben aber die Qual der Wahl. Drei Buchsen befinden sich dort. Aber welche ist die richtige? Unter der größten steht "TTL.RGB". An sie kann, wie bereits gesagt, ein PC angeschlossen werden. Links davon finden wir zwei Cinch-Buchsen. Sie sind mit "CVBS IN" und "AUDIO IN" beschriftet. Das Wörtchen "IN" besagt, daß es sich hier um Eingänge handelt. Audio dürfte auch klar sein. Über diesen Eingang werden die Töne zugeführt. Bleibt nur noch "CVBS" übrig. Das muß dann wohl die Buchse sein, über die das Bild eingespeist werden muß. Nun hängt es von Ihrem Rekorder ab, mit welchen Adaptern Sie die Verbindung herstellen. Jedenfalls müssen Sie an einem Ende einen Cinch-Stecker für das Bild und einen weiteren für den Ton haben.

Also: Sie haben ein Kabel gekauft, das auf der einen Seite an den Rekorder und an der anderen an den Monitor paßt. Nachdem alles richtig angeschlossen ist, können wir die ersten Filme sehen - gäbe es da nicht einige Problemchen. Wir müssen uns z.B. noch überlegen, wie wir den richtigen Fersehkanal auswählen. Auch in bezug auf die Lautstärke werden wir uns in den meisten Fällen noch ein paar Gedanken machen müssen. Zunächst wollen wir uns jedoch die Grundlagen vornehmen.

Um das Bild, das am Video-Eingang anliegt, auch sehen zu können, müssen wir die Klappe an der Frontseite des Monitors öffnen, hinter der sich neben einigen Reglern noch zwei Schalter verbergen. Mit dem rechten dieser Schalter kann man festlegen, ob man ein farbiges Bild haben möchte oder ob man (aus welchen Gründen auch immer) lieber einer grünen Darstellung den Vorzug gibt. Links daneben liegt ein weiterer Schalter. Mit ihm können wir zwischen CVBS (Schalter in gedrückter Stellung) und RGB (Schalter in nicht gedrückter Stellung) wählen. Wir entscheiden uns für CVBS, wodurch nun das Bild angezeigt wird, das wir über unser Kabel zuführen.

Wenn Sie nun den Rekorder einschalten, können Sie in den meisten Fällen ein Bild sehen. Eigentlich kann man an jedem Video-Gerät, das nicht zur Reihe der totalen Billigfabrikate gehört, einstellen, welches Programm über den Ausgang des Rekorders ausgegeben wird, wenn dieser nicht gerade einen Film wiedergibt. Wenn Sie nicht wissen, wie das geht, können Sie entweder so lange probieren, bis es klappt, das Handbuch lesen oder einen informierten(!) Fachhändler aufsuchen, um sich diesbezüglich helfen zu lassen. Spätestens dann müßten Sie aber etwas sehen, wenn Sie einen Film starten.

Das Bild ist da - aber hören Sie auch etwas? Damit sprechen wir einen etwas heiklen Punkt an. Wenn Sie nämlich etwas hören, wird es vermutlich ziemlich laut sein. Bevor Sie sich jetzt aber bemühen, den Lautstärkeregler des Monitors nachzustellen, sollten Sie sich überlegen, ob es nicht günstig wäre, einen eigenen Regler für den Video-Rekorder zu installieren, mit dem man die Lautstärke anpassen kann. Wir können Ihnen diese Methode nur wärmstens empfehlen, da Sie sonst immer, wenn Sie etwas mit dem Amiga machen wollen, den Regler des Monitors wieder zurückdrehen müssen. Die Ursache hierfür begründet sich darin, daß der Amiga seine Töne etwas leiser von sich gibt, als das ein Rekorder oder Hifi-Gerät tut. Um diese Differenz auszugleichen, kann man in die Ton-Leitung des Rekorders einen regelbaren Widerstand einbauen. Man bezeichnet einen solchen übrigens auch als Potentiometer oder Poti. Wie man ihn auch nennt, seine Funktion bleibt es, eine Spannung um einen gewissen Pronzentsatz zu drosseln. In unserem konkreten Fall bedeutet das nichts anderes, als daß die Lautstärke reduziert wird.

Der Einbau des Potentiometers ist nicht besonders schwierig. Man muß es einfach in die Leitung einbauen, die an die Tonbuchse angeschlossen wird. Leider können wir Ihnen nicht jeden nötigen Handgriff beschreiben. Sollten Sie daher mit dem Einbau Schwierigkeiten haben, wenden Sie sich bitte, an einen Hobbybastler - für ihn ist es wahrscheinlich kein Problem. Beim Einbau muß man eigentlich nur zwei Dinge beachten. Man muß erstmal die richtigen Pins des Reglers erwischen. Im allgemeinen gibt es dabei eine einfache Regel. Man nehme den mittleren und einen der äußeren Pins, wenn drei vorhanden sind. Regel Nummer zwei ist wichtig, um ein sehr lautes Hintergrundrauschen zu vermeiden, das auch die lautesten Klänge übertönt. Achten Sie darauf, daß Sie die scheinbar unnötige Metallfaserummantelung des Audio-Kabels nicht unterbrechen. Läßt sich dies nicht vermeiden, müssen Sie sie mit etwas Lötzinn wieder zusammenflicken.

Bevor Sie allerdings beginnen, das Poti in eine Leitung einzusetzen, müssen Sie erst einmal feststellen, welche der Leitungen überhaupt die richtige ist. Bei einem Mono-Monitor müssen Sie zwei der meistens vierfach vorhandenen Cinch-Stecker zusammenführen. Welche die richtigen sind, kann man leicht ausprobieren. Auf die Angaben, die der Hersteller zu dem Kabel macht, kann man sich oft nicht verlassen, da es oft verschiedene Auslegungsmöglichkeiten gibt.

Nennen Sie den 1084er oder einen anderen Stero-Monitor ihr eigen, müssen Sie die beiden Tonkanäle natürlich nicht zusammenführen. Sie können Sie an

die vorgesehenen Anschlüsse einstecken und damit in den Genuß einer Stereo-Wiedergabe kommen. Möchten Sie die Lautstärke drosseln, müssen Sie entweder ein Poti in jede Leitung einbauen oder ein Doppel-Poti verwenden, das mit einem Regler zwei Leitungen schaltet. Man kann solche Potis in Elektronik-Geschäften bekommen.

Über einen TV-Tuner

Wer keinen Video-Rekorder besitzt oder wer seinen Rekorder nicht im gleichen Zimmer stehen hat wie den Amiga, kann seinen Monitor trotzdem als Fernseher einsetzen. Er muß allerdings ca. 200 DM in einen TV-Tuner investieren. Ein solches Gerät ist sozusagen ein Fernseher ohne Bildschirm. Es wird an den Video-Eingang des Monitors angeschlossen, auf dem es dann das Fersehbild darstellt. Natürlich kann man bei einem TV-Tuner mehrere Kanäle wählen. Übrigens können Sie den 1081 oder 1084 auch an einen C64 anschließen. Dabei müssen Sie allerdings darauf achten, daß Sie ein Kabel kaufen, das auch die Farbinformation richtig überträgt (also keins, das für den Monitor 1702 vorgesehen ist). Es wird an den Cinch-Buchsen des Monitors und am Monitor-Ausgang des C64 angeschlossen. Allerdings sollte man die Audio-Leitung mit einem Widerstand versehen, da diese Leitung mit relativ hohen Pegeln arbeitet.

4.6.3 Schönheitsoperation am 1081

Im letzten Kapitel erzählten wir ihnen, daß man den Monitor 1081 sowohl an den Amiga als auch an ein Video-Signal (z.B. von einem TV-Tuner) anschließen kann. Das ist prinzipiell auch richtig, aber jetzt kommt der Haken. Möchte man das Fernsehbild sehen, muß man den Amiga ausschalten, um ein sauberes Bild zu erhalten. Andernfalls sehen Sie nichts vernünftiges. Dieser Umstand kann unter Umständen sehr lästig werden. Mit einem kleinen Trick kann man diesen Fehler aber umgehen. Dazu muß man nicht einmal löten. Es reicht vollkommen aus, ein Bauteil von der Platine des Monitors zu entfernen. Bei diesem Bauteil handelt es sich um einen Widerstand, dem man die Ordnungszahl R504 auf der Platine zugeteilt hat. Kneift man ihn durch, kann man ohne Bildstörung beide Signalquellen nebeneinander verwenden. Allerdings hat diese Veränderung auch einen kleinen Nachteil. R504 hat nämlich nicht die Aufgabe, das Bild des Cinch-Eingangs zu stören, was nur ein Nebeneffekt ist,

sondern ist dazu da, den Anschluß eines Video-Rekorders am Scart-Eingang zu ermöglichen. Aber wer macht das schon?

Wer den Scart-Eingang ab und zu für den Anschluß eines Rekorders braucht und andererseits auch ein gutes Bild über die Cinch-Eingänge haben möchte, wenn der Amiga eingeschaltet ist, braucht deshalb nicht enttäuscht zu sein. Man kann nämlich einen Schalter am Widerstand installieren. Dazu muß man dann allerdings löten (oder löten lassen).

Alles klar? Dann kann's ja losgehen. Bei den nun folgenden Arbeiten sollten Sie sich immer vor Augen führen, daß es nicht ungefährlich ist, an einem Monitor herumzubasteln. Neben der Netzspannung, die man dadurch vermeiden kann, daß man den Stecker aus des Dose zieht, was eigentlich selbstverständlich ist, ist noch eine weitere Spannung vorhanden, die weit gefährlicher sein kann, als die Netzspannung. Gemeint ist der zum Betrieb der Bildröhre notwendige Strom, der auch noch lange nach dem Ausschalten des Geräts vorhanden ist. Überlegen Sie sich daher, einen Fachmann den Umbau vornehmen zu lassen. Wenn Sie jedoch einigermaßen geschickt sind, können Sie ihn auch problemlos selbst durchführen.

Zunächst ist das Gehäuse des Monitors zu entfernen. Es wird von vier Schrauben gehalten, von denen sich zwei unter dem Gerät und zwei auf dessen Oberseite befinden. Nachdem man sie gelöst hat, kann man das große Plastikteil nach hinten abziehen. Beachten sollte man dabei allerdings, daß der Lautsprecher daran befestigt ist. Er ist mit einem kleinen Stecker mit der Platine verbunden. Hat man ihn herausgezogen, kann man das Teil entfernen. Als wir unseren Monitor zum ersten Mal auseinandernahmen, wußten wir nicht, daß es dieses Steckerchen gibt. Wir demontierten also den Lautsprecher direkt. Von dieser Methode können wir Ihnen aber nur abraten - ein Teil blieb beim späteren Zusammensetzen nämlich übrig!

Der gesuchte Widerstand befindet sich in der Nähe der Cinch-Eingänge. Sie müssen allerdings etwas suchen, um ihn zu finden. Haben Sie ihn lokalisiert, können Sie ihn auf einer Seite durchknipsen. Dabei sollten Sie allerdings nicht den Keramikteil, sondern eine Zuleitung trennen. So stellen Sie sicher, daß Sie zu einem späteren Zeitpunkt einen Schalter montieren können, um die Verbindung je nach Bedarf wieder herzustellen.

Nun können Sie den 1081 wieder zusammenbauen.

4.6.4 Mehr Rechenleistung für wenig Geld

Der Amiga ist einer der schnellsten Rechner im Heim- und PC-Bereich. Das ist eine Tatsache, die allgemein bekannt ist. Durch diverse Zusätze kann man seine Geschwindigkeit aber noch erheblich steigern. In diesem Zusammhang fallen dann meisten Bezeichnungen wie 68020, 68030 und 68881. Hinter ihnen verbergen sich Prozessoren, die sowohl leistungsstark als auch teuer sind. Für Profis ist ihre Anschaffung lohnend. Doch welcher Anwender gibt schon gerne tausende von Mark für einen Luxus aus, den er doch nicht richtig einsetzen kann?

Nun gibt es aber einen Prozessor, der etwas schneller ist als der MC68000 und dazu noch für ca. 40-80 DM zu haben ist. Gemeint ist der MC68010. Er ist pinnkompatibel zum 68000er, was nichts anderes bedeutet, als daß man beide Prozessoren gegeneinander austauschen kann. Bevor Sie nun aber das Buch fallen lassen, um zum nächsten Elektronikhändler zu rennen, müssen wir Ihnen allerdings noch auf eine kleine Schwäche des MC68010 hinweisen. Er ist nämlich nicht vollständig kompatibel zum alten 68000er. Ein Maschinenbefehl funktionert nämlich etwas anders, als das früher der Fall war. Folge: Systemabsturz!

Mit diesem Problem kann man aber leben, da der Befehl nur von wenigen Programmen genutzt wird. Es gibt allerdings noch eine Routine im Kickstart 1.2, die sich mit dem 68010 auch nicht verträgt. Wer mit Programmen arbeitet, die diese Routine oder den anders arbeitenden Befehl verwenden, kann sich dank der offenen Architektur der 68000er-Reihe mit einem kleinen Programm behelfen, das sich auf der Public-Domain-Diskette "Fish 18" befindet. Bald wird auch der Profimat Amiga von DATA BECKER mit einem ähnlichen Programm ausgeliefert werden.

Für Assembler-Programmierer dürfte auch der erweiterte Register- und Befehlssatz des MC68010 interessant sein. Besonders die Abarbeitung von kurzen Schleifen wird erheblich beschleunigt. Reine Anwender werden sich allerdings nur über die etwas höhere Geschwindigkeit freuen. Man sollte allerdings nicht erwarten, daß der Amiga doppelt so schnell arbeitet. Eine Steigerung von 10% kann man aber als realistisch bezeichnen.

Zum Einbau des Prozessors ist eigentlich nicht viel zu sagen. Sie müssen dazu lediglich den Amiga öffnen, wie in Kapitel 4.1.2 beschrieben. Dann ist der fette 68000er aus seinem Sockel zu wuchten. Dabei sollte man eine gewisse

Vorsicht walten lassen, um ihn nicht zu beschädigen. Vielleicht will man ihn später ja noch einmal verwenden. Beschädigungen können eigentlich nur durch zwei Dinge auftreten. Entweder bricht man einen Pin ab, oder zerstört das empfindliche Innenleben durch eine statische Entladung. Uns ist zwar noch kein Chip auf diese Weise kaputtgegangen, dennoch besteht ein gewisses Risiko, das man aber umgehen kann, indem man immer eine Hand erdet. Anfassen sollte man die Pins natürlich nicht direkt.

Um den auf 64 Beinen stehenden Prozessor aus seinem Sockel zu nehmen kann man entweder eine spezielle Chip-Zange in der richtigen Größe oder einen Schaubenzieher verwenden, den man zwischen Prozessor und Sockel schiebt. Hüten Sie sich aber davor, mit dem Schraubenzieher zu hebeln, da sonst die Platine Schaden nehmen könnte. Besser geht es durch eine leichte Drehung. Setzt man abwechselnd oben und unten an, kann man den Chip so weit lockern, daß man ihn vorsichtig herausziehen kann. Der Schraubenzieher muß übrigens immer an den kurzen Seiten des Prozessor angesetzt werden, da die anderen ja mit unzähligen Pins bestückt sind, die den Einwirkungen eines Schraubenziehers natürlich nicht gewachsen wären.

Merken Sie sich bitte, in welche Richtung die Kerbe des Chips zeigte. Die Kerbe des 68010 muß nämlich bei dessen Einbau in die gleiche Richtung zeigen. Beim Einbau sollte man noch mehr Vorsicht walten lassen, als beim Ausbau. Auch hier kann man wieder eine Chip-Zange verwenden. Man kann das gute Stück aber auch durch leichten Druck von Hand einsetzen. Vorher sollte man allerdings sicherstellen, daß alle Pins an der richtigen Stelle sitzen. Andernfalls kann es leicht passieren, daß sich der 68010 ein Bein bricht und dadurch arbeitsunfähig wird. Oft werden Chips auch mit stark abstehenden Beinchen geliefert. Bevor man sie einsetzen kann, muß man die Pins erst gerade biegen. Dazu kann man den Chip entweder seitwärts auf eine ebene, nicht leitende Fläche drücken, oder mit einem speziellem Werkzeug bearbeiten, das man in Elektronikläden und -versänden findet. Bei diesem Wekzeug handelt es sich wieder um eine Zange, mit der man einen Chip sockelgerecht herrichten kann. Auf keinen Fall dürfen Sie die Pins einzeln mit den Fingern richten!

Nach vollzogenem Umbau können Sie den Amiga wieder zusammenbauen. Sollte er jetzt nicht funktionieren, können mehrer Fehlerquellen vorliegen. So kann es zum Beispiel sein, daß Sie einen der Stecker nicht oder nicht richtig angeschlossen haben. Auch sollten Sie überprüfen, ob der eingebaute Prozessor richtig sitzt. Ist das der Fall, wird es schon kritischer. Entweder ist der

68010 defekt, oder der Amiga hat Schaden erlitten. Den ersten Fall können Sie leicht überprüfen, indem Sie den 68000er wieder einsetzen. Im allgemeinen geht aber alles glatt.

4.6.5 Nullmodem im Eigenbau

In Kapitel 4.5.1 stellten wir Ihnen ein Nullmodem vor. Nun ist es an der Zeit, zu beschreiben, wie man ein solches baut. Dabei gibt es verschiedene Techniken. Je nachdem, welche Übertragungsparameter man einstellt, muß das Kabel eine bestimmte Anzahl von Leitungen haben. Oft verwendet man ein dreiadriges Übertragungssystem, das jedoch nicht alle Möglichkeiten bietet, die man unter Umständen brauchen könnte (Übertragung stoppen usw.). Wir werden ihnen daher ein Kabel vorstellen, das die Daten über sieben Leitungen übermittelt.

Um dieses Kabel herzustellen, benötigen wir zwei 25polige RS232C-Buchsen. Man bezeichnet sie auch gerne als SUB-D-Buchsen. Weiterhin ist es ratsam, die dazu passenden Gehäuse zu kaufen. Dabei gibt es verschiedene Ausführugen, die sich nicht nur im Preis, sondern auch in der Ausführung unterscheiden. Wir empfehlen Ihnen solche, die man (ähnlich wie den Drucker- und RGB-Stecker) mit dem Gerät verschrauben kann. Nun müssen wir uns noch ein geeignetes Kabel besorgen. Hier bietet sich auch ein Flachbandkabel an. Wenn wir alles besorgt haben, heizen wir den Lötkolben schon mal ein. Wenn er warm ist, kann's losgehen. Auf den Buchsen sind meistens kleine Zahlen aufgedruckt, die die Pins numerieren. Uns interessieren dabei nur die folgenden Kontakte:

Pin	Abkürzung	Name	Funktion
2	TXD	Transmitted data	Daten
3	RXD	Received data	
4	RTS	Ready to send	Handshake
5	CTS	Clear to send	
6	DTR	Data set ready	
20	DSR	Data terminal ready	
7	GND	Ground	Masse

Die Verbindungen sind folgendermaßen herzustellen:

Stecker 1	Stecker 2
2	3
3	2
4	5
5	4
6	20
20	6
7	7

4.6.6 Ein Schalter für den Speicher

Wir haben es bereits an vielen Stellen in diesem Buch angekündigt: Mittels einer kleinen Bastelei kann man einen Schalter installieren, der es ermöglicht, das zusätzliche Fast-RAM einer internen Speicherkarte abzuschalten.

Diesen Schalter kann man an drei verschiedenen Stellen anbringen: direkt auf der Platine der Speicherkarte, an einem Zwischenstück, das zwischen die Karte und den Amiga-Steckplatz gesteckt wird, oder direkt an der Amiga-Platine. Hier würde sich ein Zwischen-Sockel für Gary empfehlen, da die zu unterbrechende Leitung dort mündet.

Da es ziemlich schwierig ist, die Veränderung direkt auf der Platine der Speicherkarte vorzunehmen, haben wir uns entschieden, den Bau eines Zwischenstücks zu beschreiben.

Wem es auf 29,90 DM nicht ankommt, kann diese Abschaltung auch direkt bei Message Computer bestellen. Es kann nämlich ziemlich schwierig sein, die passenden Stecker- und Buchsenleisten zu besorgen.

Für die Herstellung des Zwischenstücks benötigen Sie eine Buchsen-Leiste (2x28 Pole; Rastermaß 2.54; gerade Pins), eine dazu passende Stecker-Leiste, einen halben Meter zweiadriges Kabel, Lötzinn, Löterfahrung und einen kleinen Schalter. Die Buchsen- und Stecker-Leisten werden nun zusammengelö-

tet. Nur Pin 31 wird dabei ausgelassen. Doch wo liegt dieser Pin? Die folgende Grafik versucht dies zu verdeutlichen:

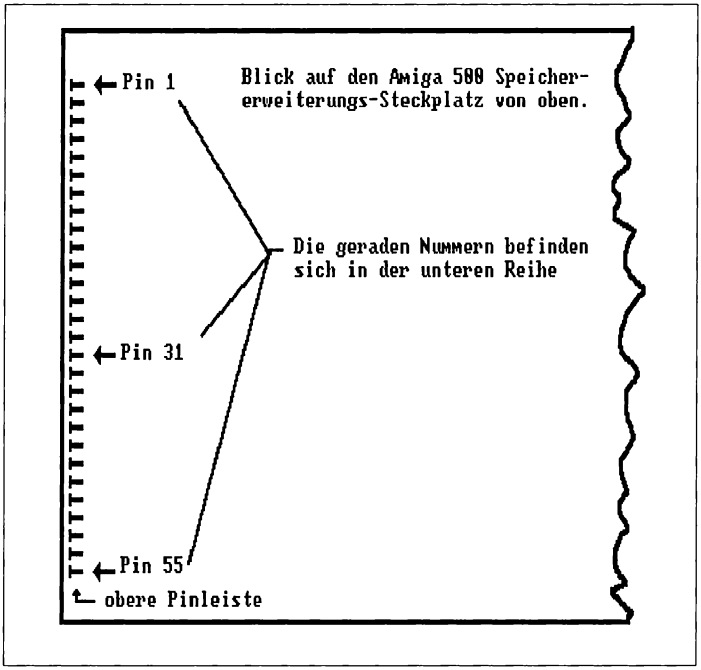


Abb. 4.2: Kontaktleiste

An die beiden Kontakte, die bei Pin 31 nicht miteinander verlötet wurden, wird je eine Ader des Kabels angelötet. Am anderen Ende der Leitung wird der Schalter installiert. Wenn Sie dies getan haben, können Sie den Speicher-Ausschalt-Adapter in den Amiga einbauen. Achten Sie darauf, daß Sie ihn so einstecken, daß die Pin-Reihe, in die der Schalter eingebaut wurde, der Tastaturseite des Amiga näher ist.

4.6.7 Bau einer Joystick-Weiche

An der Rückseite des Amiga finden wir zwei kleine SUB-D-Buchsen, die mit "1.JOYSTICK" und "2.JOYSTICK" beschriftet sind. An einem dieser An-

schlüsse ist die Maus angeschlossen, am anderen... ja, am anderen kann alles mögliche angeschlossen sein: ein Joystick (Steuerknüppel), ein Sound-Digitalisierer und vieles mehr. Da es auch Spiele gibt, die mit 2 Joysticks gespielt werden können oder müssen, wird man oft die Stecker wechseln müssen.

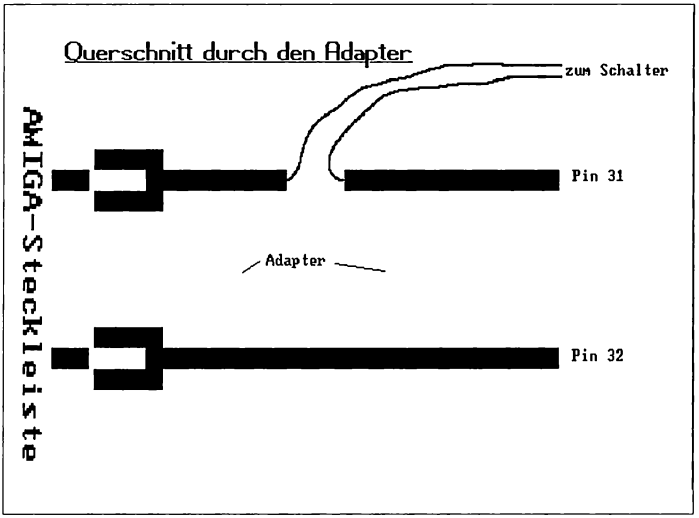


Abb. 4.3: Querschnitt durch den Adapter

Daß das nicht gerade angenehm ist (für Sie und die Stecker) müßte eigentlich jeder einsehen. Was ist also zu tun? Ganz einfach: eine Joystick-Weiche muß gekauft oder gebaut werden. Ein solches Gerät (Gerät ist schon übertrieben) ist nichts anderes als ein Kabel, dessen eines Ende an den Amiga angeschlossen wird. Am anderen Ende können nun aber nicht ein einziges, sondern gleich zwei (oder noch mehr) Eingabegeräte angeschlossen werden.

Auf diese Weise kann man an beiden Ports je einen Joystick und an Port 1 zusätzlich eine Maus bzw. an Port 2 einen Digitalisierer oder etwas anderes anschließen.

Joystick-Weichen kann man in zwei Ausführungen kaufen: als relativ billige, einfache Verzweigung ohne und als komplexere und damit auch teurere (aber kein bißchen sinnvollere) Ausführung mit Umschalter. Wir haben mit einer Version ohne Umschalter gearbeitet und keine Probleme damit feststellen

können. Wer Geld sparen möchte, kann sich seine Weichen natürlich auch selbst herstellen. Dazu benötigt man

- 1m 9-adriges Kabel
- eine 9-polige SUB-D-Kupplung (mit Gehäuse)
- zwei oder mehr 9-polige SUB-D-Stecker (mit Gehäuse)
- Lötgerät und ein bißchen Löterfahrung
- eventuell einen Umschalter (zur Not für alle Leitungen)

4.7 500er-Tuning

Da Sie im Besitz eines Amiga 500 sind oder wenigstens beabsichtigen, sich einen solchen zu kaufen, können Sie normalerweise keine Erweiterungen verwenden, die für den Amiga 2000 entwickelt wurden und für den internen Einbau in einen der Steckplätze vorgesehen sind.

Aufgrund der Tatsache, daß der 2000er relativ teuer ist, hat die Firma Comptec eine Marktlücke geschlossen: sie stellt einen Bausatz her, mit dem es möglich ist, einen Amiga 500 um einige 2000er-Steckplätze zu erweitern. Dazu muß man den 500er zerlegen und anschließend in ein Gehäuse einbauen, das neben dem "alten internen" noch ein weiteres 3½-Zoll-Laufwerk aufnehmen kann. Die Tastatur wird, ähnlich wie bei einem PC oder einem Amiga 2000, von der Zentraleinheit abgesetzt installiert. Die Kosten für dieses Gehäuse belaufen sich auf 398,- DM.

Leider ist es damit aber noch nicht getan. Will man 2000er-Hardware verwenden, muß man noch 298,- DM (und eventuell 29,- Mark für den wirklich leisen Lüfter) auf den Tisch blättern. Dann hat man jedoch einen fast vollwertigen 2000er (ohne MMU-Kartenslot, Videoslot etc.). Der Witz bei der Sache ist, daß man, wenn man einen 500er in einen "2000er" umfunktioniert, weniger Geld ausgeben muß, als wenn man gleich einen richtigen Amiga 2000 kaufen würde.

Alle die, die sich Ihren 500er noch nicht gekauft haben, können bei der Herstellerfirma des 500er-Tuning-Bausatzes auch ein fertig montiertes Gesamtpaket kaufen, bei dem in das Tuning-Gehäuse bereits die Amiga-Hardware eingebaut wurde. Man erspart sich so die lästige Montagearbeit, die sicherlich

nicht jedermanns Sache ist. Auch wenn man bereits einen 500er hat, muß man diesen nicht unbedingt selbst einbauen. Comptec nimmt den Käufer auch diese Arbeit ab. Sie müssen dazu lediglich Ihren Amiga einschicken. Allerdings werden Ihnen für diesen Service 50,- Mark extra berechnet. Diesen Betrag sollten Sie jedoch investieren, da beim Selbsteinbau immer unvorhersehbare Probleme auftreten können, die z.B. durch unterschiedliche Amiga-Baureihen bedingt sein können.

Die Hardware

Das Gehäuse des Tuning-Bausatzes ist so konzipiert, daß es

a) als Monitor-Ständer verwendent werden kann.

b) die Möglichkeit bietet, die abgesetzte Tastatur unter ihm verschwinden zu lassen.

c) zwei 3½-Zoll-Laufwerke aufnehmen kann.

d) Platz für vier Amiga- oder PC-kompatible Karten bietet. Allerdings gibt es an der Gehäuserückwand nur drei Öffnungen, weshalb maximal drei Karten mit externen Schnittstellen verwendet werden können. Sehr gut fanden wir, daß alle vier Steckplätze auch mit AT-Karten bestückt werden können. Im original 2000er sind es nämlich nur zwei. Baut man also eine AT-Brückenkarte ein, bleiben noch drei Steckplätze frei, die mit AT-kompatiblen Karten bestückt werden können.

Auch optisch steht das Gerät von Comptec seinen Mann.

Ein letzter Tip für Profi-Bastler

Möchten Sie das Geld für das Tuning-Gehäuse sparen, so reicht es aus, den Adapter für die Aufnahme der 2000er-Karten zu kaufen. Dieser kann dann direkt an den Expansionsport des Amiga angesteckt werden. Auch eine Verbindung über ein selbstgebautes Flachbandkabel wäre denkbar (und sinnvoll). Entscheidet man sich für diesen Weg, wird man ein eigenes Gehäuse um den

Adapter bauen müssen, da die Karten sonst keinen Halt finden und damit leicht beschädigt werden.

Das Netzteil des Adapters ist so angebaut, daß es, wenn man den eben angeregten Weg wählt, von der Adapter-Platine abmontiert werden muß. Das geht ganz einfach.

Sie sollten, wenn Sie diese Idee verwirklichen wollen, wirklich ein sehr guter Bastler sein!!!

5. Vom Anwender zum Programmierer

Nach einiger Zeit der Software-Nutzung strebt man zu größeren Dingen. Nicht jede Gegebenheit wird so akzeptiert, wie sie eine gekaufte Software vorstellt. Eigene Vorstellungen der Programmgestaltung werden laut, die aber nicht in die Tat umgesetzt werden können. Das ist die Stunde der Entscheidung! Man beschließt, eigene Programme zu schreiben.

Die richtige und zweckmäßige Programmiersprache wird gesucht. Welche Sprache ist einfach? Welche Sprache ist besonders sinnvoll? Welche Sprache ist kompatibel? Und welche Sprache kann ich persönlich leicht erlernen?

Viele Fragen stellen sich dem Anwender beim Schritt zum Programmierer. Wir werden jetzt hier anhand von Beispielen in den drei wichtigsten Programmiersprachen auf dem Amiga, nämlich BASIC, Assembler und C den Einstieg in die Programmierung des Systems wagen, um zu zeigen, was alles mit welcher Sprache möglich ist.

Die Sprache BASIC hat ihren besonderen Reiz dadurch erhalten, daß sie durch einen an die natürliche Sprache angelegten Wortschatz besonders leicht zu erlernen ist. Weiterhin hat sie auf dem Amiga eine ganz besondere Bedeutung, weil sie kostenlos im Gesamtpaket enthalten ist.

An dieser Stelle wollen wir keine Einführung in die Programmiersprache BASIC bieten. Dies haben unsere Kollegen Hannes Rügheimer und Christian Spanik mit ihrem überaus erfolgreichen Buch "AMIGA-BASIC" schon getan. Dieses Kapitel soll vielmehr zeigen, wie man über das AmigaBASIC sein gesamtes Betriebssystem nutzen kann. Wir werden zu allen wichtigen Gebieten (Grafik, Floppy, Systemprogrammierung) viele kleine Beispielprogramme vorstellen, so daß Sie nicht nur wissenswertes zur Programmierung erfahren, sondern auch den Einstieg in die Betriebssystem-Programmierung geboten bekommen.

Als Einstieg möchten wir zuerst mit einem reinen BASIC-Programm aufwarten, das zum Berechnen der bekannten Apfelmännchen genutzt werden kann. Es baut die Grafik im normalen Ausgabe-Fenster des AmigaBASIC Punkt für

Punkt auf. Sehen Sie sich zuerst das Listing an, an dem wir dann das Programm besprechen können:

```
"************************************
"*                                  *
"* Programm: Apfelmännchen          *
"*                                  *
"************************************
AnzFa% = 4
MaxFa% = AnzFa%-1
DEF FN Color(x)=x-INT(x/MaxFa%)*MaxFa%
xo = 10     : yo =   5     " Zeichenbereich im
xg = 600    : yg = 150     " Ausgabe-Window
x1 = -2.25 : y1 = -1.75    " Eckkoordinaten des
x2 =  1     : y2 =  1.75   " Ausschnitts
Tiefe% = 4                 " Berechnungstiefe (Anzahl
                           " der Durchläufe)
dx = (x2-x1)               " Breite und Höhe des
dy = (y2-y1)               " Ausschnitts

FOR y = 0 TO yg-1 STEP 10
  b = y1+dy*y/yg
  FOR x = 0 TO xg-1 STEP 10
    a = x1+dx*x/xg
    Wert = 0
    C1 = .1
    C2 = .5
    Flag% = 1
    Z% = 0
    WHILE Wert < AnzFa% AND Flag% = 1
      CRW1 = C1
      CRW2 = C2
      C1 = CRW1*CRW1-CRW2*CRW2+a
      C2 = 2*CRW1*CRW2+b
      Wert = C1*C1+C2*C2
      Z% = Z%+1
      If Z% = Tiefe% THEN Flag% = 0
    WEND
    If Z% = Tiefe% THEN
      PSET (x,y), 0
    Else
      PSET (x,y), FN Color(Z%)+1
    END If
  NEXT x
NEXT y
```

Listing 5.1: Apfelmännchen

Programmbeschreibung

Wie Sie am Anfang des Listings feststellen können, beginnt dieses mit einigen Variablen-Definitionen. Diese stellen alle wichtigen Werte für das Programm ein. Deswegen sollten wir darauf einen aufmerksamen Blick werfen.

Mit AnzFa% und MaxFa% legen wir die Anzahl der Farben fest. Auf der normalen Workbench-Screen stehen uns wegen der zwei BitPlanes vier Farben zur Verfügung. Dementsprechend steht in MaxFa% auch der Wert Drei, da die größte Farbnummer immer um Eins vermindert ist. Das liegt daran, daß auch die Zahl Null benutzt wird.

Die darauf definierte Funktion wird später zur Bestimmung der Farbe eines Punktes benutzt. Je nach der Anzahl der Durchläufe innerhalb der Berechnungs-Schleife bemißt sich ja die Farbwahl. Das übernimmt die Funktion in Abhängigkeit von der maximalen Farbanzahl. Sollten Sie also ein Window auf einer Screen mit mehr Farben eröffnen, müssen Sie nur die Variable AnzFa% entsprechend anpassen, und alle Berechnungen, die damit in Zusammenhang stehen, werden angepaßt.

Als nächstes ist es nötig, den Bereich im Ausgabe-Window festzulegen, in dem die Apfelmännchen-Grafik ausgegeben werden soll. Vielleicht wollen Sie nicht das gesamte Window nutzen, weil noch Texte oder Zahlen hineingeschrieben werden sollen. Besonders bei einer so zeitaufwendigen Berechnung ist es wichtig, bei jedem unnötigen Pixel darauf zu achten, daß es nicht gezeichnet wird. Man spart dadurch einige Sekunden! Das Variablen-Paar xo, yo speichert den Origin-Wert, also den Ausgangspunkt, der bei Windows bekanntlich in der linken oberen Ecke liegt. Mit den zweiten Koordinaten geben wir den anderen Eckpunkt der Ausgabe-Fläche an.

Weiterhin werden neben diesen Koordinaten für die Ausgabe noch die Koordinaten der Berechnungen gebraucht. Denn die Berechnungen in der Menge der komplexen Zahlen, in der wir uns hier ja befinden, um Apfelmännchen zu konstruieren, benötigen immer zwei Werte, als befände man sich auf einem Koordinatensystem. Hier ist mit den Einstellungen ein bekannter Ausschnitt angewählt, der unter Garantie ein Standard-Apfelmännchen liefert. Verkleinern Sie diese Werte nach mehrmaligen Versuchen ruhig um einiges, um sich Ausschnitte anzusehen, die viele interessante Details beinhalten.

Die Variable Tiefe% legt eine maximale Verschachtelung für die späteren Berechnungen fest. Je größer der hier eingestellte Wert ist, desto differenzierter wird die Grafik. Allerdings gibt es ab einer bestimmten Größe keine wirklichen Verbesserungen mehr. Sie erhöhen dadurch nur unnötig die Berechnungszeit und verlangsamen dadurch die Ausgabe erheblich.

Als letzte Variablen werden die Breite und die Höhe des grafischen Ausschnitts berechnet, um bei der Übertragung der beiden Koordinatensysteme - das der komplexen Zahlen und das der Grafikausgabe - ein Verhältnis zu haben.

Kommen wir nun zur eigentlichen Berechnung und grafischen Ausgabe. Diese wird umrahmt vom zwei Schleifen, die systematisch die einzelnen Punkte der Grafik ablaufen. Die äußere geht dabei die Y-Koordinaten durch, die innere die X-Werte. Beide sind in dem abgedruckten Programm mit einer Schrittweite von 10 eingestellt. Das erzeugt natürlich bei der Ausgabe mehr ein Punkt-Raster als ein Apfelmännchen. Sie können damit aber überprüfen, ob das Programm funktioniert, und müssen nicht erst einige Minuten warten, bis Sie das Ergebnis vor sich sehen.

Zu Anfang der beiden Schleifen werden Grundwerte berechnet, die den jeweiligen Basis-Betrag darstellen. Dann werden die Variablen für die eigentliche Berechnung initialisiert; zwei Zahlen aus dem komplexen Bereich, ein Zähler und die Flagge, die das Erreichen der Obergrenze kennzeichnet. Die Schleife selbst führt immer wieder die Quadrierung beider Zahlen innerhalb der Menge der Komplexen Zahlen durch. Für jeden Rechenschritt wird der Zähler um eins erhöht. Hat er das Maximum erreicht oder ist der errechnete Wert größer als die Anzahl der Farben, wird die Schleife abgebrochen.

Nun kann der Punkt grafisch ausgegeben werden. Dafür muß das Abbruchkriterium untersucht werden. Wurde nur abgebrochen, weil die Berechnungstiefe erreicht wurde, heißt die Zeichenfarbe 0. In jedem anderen Fall kann die Zeichenfarbe durch die am Anfang definierte Funktion berechnet werden. Diese ermittelt aus der benötigten Tiefe einen Farbwert im erlaubten Bereich. Fertig!

Wenn diese gesamten Berechnungen genügend oft durchlaufen wurden, steht einem gelungenen Apfelmännchen nichts mehr im Wege!

5.1 Alles nutzen: Systemroutinen

Sicherlich bietet das AmigaBASIC mit seinem großen Befehlssatz ein breites Spektrum der Anwendungsmöglichkeiten. Aber leider läßt sich nicht wirklich alles damit machen. Geschwindigkeit und Komplexität spielen hier ein Rolle, die sich von BASIC aus nicht erreichen läßt.

Deshalb haben sich die Programmierer von Microsoft, der Software-Firma, die AmigaBASIC programmiert hat, gleich von Anfang an eine Tür offen gelassen, die es ermöglicht, alle Routinen, die im Amiga-Betriebssystem vorhanden sind, zu nutzen.

Diese fantastische Möglichkeit, auf alles zugreifen zu können, was sich im Betriebssystem befindet, eröffnet ganz neue Wege für die Programmierung, von denen man bisher gar nicht geträumt hatte.

Dieser Teil des AmigaBASIC-Kapitels will sich nun damit beschäftigen, wie man diese Routinen aufruft, wie diese Routinen überhaupt aufgebaut und gegliedert sind und was man bei der Arbeit beachten muß, damit man in Einklang zum System programmiert.

5.1.1 Der Aufbau des Systems

Schon bei der Entwicklung des Amiga hat man sich Gedanken über den Aufbau seines Betriebssystems gemacht. Ältere Systeme wie z.B. das des Commodore 64 waren in ein ROM gebrannt und starr. Sie konnten nur über umständliche Umwege ergänzt oder korrigiert werden. Vorhandene Fehler waren schwer zu verbessern. Dies wollte man beim Amiga auf jeden Fall verhindern!

Deshalb hat man beim Amiga 1000 das gesamte Betriebssystem ladbar gemacht, so daß dieses nach dem Einschalten erst einmal von Diskette gebootet werden mußte. Auf der Diskette konnten gegebenenfalls noch Änderungen angebracht werden, die bei erneutem Booten dann auch für den Amiga verbindlich waren. Außerdem stellte das Betriebssystem die Möglichkeit bereit, jederzeit seinen Routinen-Umfang um einen neuen Block zu ergänzen.

Für den Amiga 500 gilt die erste Feststellung nicht mehr, denn das Betriebssystem, genannt Kickstart, ist jetzt auf einem ROM-Baustein. Dafür ist immer noch die Möglichkeit geblieben, das System um neue Funktions-Blöcke zu ergänzen. Doch bevor wir uns mit dem Gedanken vertraut machen, das Betriebssystem zu ergänzen, sollten wir erst einmal die Organisation kennenlernen.

Zur besseren Verständlichkeit eines Betriebssystems mit 192 KByte Umfang (das sind 196608 Bytes/Zeichen) wurde beschlossen, dies in kleine Gruppen zu gliedern. Dafür faßte man alle Funktionen zusammen, die sich mit der Grafik-Handhabung beschäftigen, alle, die das Diskettenlaufwerk ansteuern, und alle, die für mathematische Berechnungen zuständig sind. Dies ist natürlich nur eine Auswahl. Im ganzen gibt es dreizehn solcher Funktions-Gruppen, die in der Fachsprache Libraries genannt werden. Dieser Begriff kommt aus dem Englischen und bedeutet soviel wie Bibliotheken.

Um jetzt ganz allgemein eine dieser Funktionen nutzen zu können, muß man zuerst seinen Zugriff auf die Library anmelden, in der sich die gewünschte Funktion befindet. Es ist leider nicht möglich und sogar verboten, einfach eine Betriebssystem-Funktion ohne diese Anmeldung aufzurufen, weil wir es bei dem Amiga mit einem Multitasking-Computer zu tun haben. Jede der Befehlsgruppen läßt sich nämlich von JEDEM Programm, das zur Zeit läuft, aufrufen, und man kann sich leicht vorstellen, welches Chaos entsteht, wenn mehrere Programme auf eine Unterroutine zugreifen.

Damit wir aber nun von BASIC aus diese Funktionen nutzen können, gibt es den Befehl LIBRARY, hinter dem der Name der Library angegeben wird. Sofort wird versucht, den Zugriff auf die Library anzumelden. Da manche Libraries aber nicht fest im Speicher sitzen, sondern von der Diskette nachgeladen werden sollen, muß in manchen Fällen noch die Workbench-Diskette eingelegt werden, auf der sich die disk-residenten Libraries befinden.

Der folgende Befehl würde die Exec.Library öffnen und von BASIC aus den Zugriff erlauben:

```
LIBRARY "exec.library"
```

Bevor wir uns dem letzten Problem zuwenden, das bei der Library-Ansteuerung von AmigaBASIC aus auftritt, überlegen wir uns zuerst, wie eine Library intern organisiert ist.

Wir hatten zu Anfang gesagt, daß in einer Library bestimmte Funktionen des Betriebssystems zusammengefaßt sind. Man denkt dabei gleich an eine Unterprogramm-Sammlung, bei der ein Unterprogramm auf das andere folgt. Der Assembler-Programmierer würde sich jetzt einfach die jeweilige Adresse merken, die vergleichbar mit dem Abstand vom Anfang des Programms ist. Leider tritt bei dieser erst so praktischen Methode ein Problem auf. Wird nämlich eine Unterroutine verändert, sei es durch die Korrektur eines Fehlers oder durch die Ergänzung um eine Anwendungsmöglichkeit, würde sich der Abstand verändern, und alle älteren Programme, die sich auf die alten Abstände beziehen, könnten nicht mehr funktionieren.

Deshalb hat man eine Library etwas anders organisiert. Zu Anfang jeder Library gibt es eine Sprungtabelle, die nacheinander die Startadresse, das, was wir zuvor mit Abstand bezeichneten, enthält. Wir "springen" also nicht an den Anfang der Routine, sondern an den Punkt in der Tabelle, der für diese Routine steht. Von dort werden wir dann automatisch an die richtige Stelle gelotst.

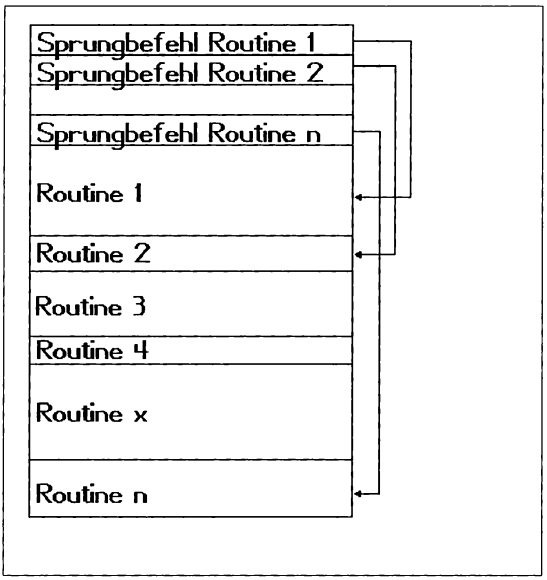


Abb. 5.1: Der Aufbau einer Library mit Sprungtabelle

Zur Programmierung und Verwendung von Libraries werden wir uns als Programmierer nur die Nummern merken müssen, die die Stelle repräsentieren,

an der die Adresse unserer Funktion steht. In der Fachsprache werden diese Nummern als Funktions-Offsets bezeichnet. Allerdings ist dieses Vorgehen etwas umständlich, schließlich gibt es mehr als 1000 Library-Funktionen, bei denen man sich unmöglich alle Offsets merken kann. Deshalb hat man für jede Funktion einen Namen eingeführt, der als Variable für den Offset steht. In der Sprache C erledigt die Übersetzung von Funktions-Name zum Offset der Compiler, in Assembler werden Tabellen eingebunden, in denen auch die Offsets definiert sind, und in AmigaBASIC gibt es .bmap-Files, die neben dem Namen alle für das BASIC notwendigen Informationen enthalten.

Ein .bmap-File ist eine Tabelle, die zuerst den Namen, dann die Parameter und zum Schluß den Offset enthält. Bei den Parametern wird ausführlich angegeben, welcher Typ von Daten angegeben werden muß und in welches Register dieser Wert übergeben werden soll. Die Register sind nur wichtig für die Assembler-Programmierer und können von uns vernachlässigt werden. Für uns sind bei jeder Funktion natürlich der Name wichtig und die Parameter, die für eine ordnungsgemäße Arbeit der Funktion notwendig sind.

Um nun ein solches .bmap-File zu erstellen, sind auf der Extras-Diskette, die zu jeder Workbench ausgeliefert wird, FD-Dateien untergebracht, in denen alle gebrauchten Informationen für eine .bmap zu finden sind. Hier ist als Beispiel ein kurzes FD-File, daß es nun auszuwerten gilt:

```
##base _IconBase
##bias 30
##public
*------ normal functions ---------------------------------
GetWBObject(name)(A0)
PutWBObject(name,object)(A0/A1)
GetIcon(name,icon,freelist)(A0/A1/A2)
PutIcon(name,icon)(A0/A1)
FreeFreeList(freelist)(A0)
FreeWBObject(WBObject)(A0)
AllocWBObject()()
AddFreeList(freelist,mem,size)(A0/A1/A2)
GetDiskObject(name)(A0)
PutDiskObject(name,diskobj)(A0,A1)
FreeDiskObject(diskobj)(A0)
FindToolType(toolTypeArray,typeName)(A0/A1)
MatchToolValue(typeString,value)(A0/A1)
BumpRevision(newname,oldname)(A0/A1)
```

Beispiel 5.1: Ein FD-File

Es handelt sich hierbei um die Aufstellung aller Funktionen, die in der Icon.Library zu finden sind. Am Anfang finden wir einige Entwickler-Informationen. Diese sind an den zwei Doppelkreuzen zu erkennen und brauchen von uns nicht beachtet zu werden.

Wichtig ist der Teil, der nach dem langen durchgezogenen Strich beginnt. Hier finden wir in jeder Zeile am Anfang den Namen der Funktion, gefolgt von den Parameter-Namen, in der ersten runden Klammer. In der zweiten Klammer stehen die Namen der Register, in denen die Parameter übergeben werden.

Da in einem .bmap-File ähnliche Informationen zu finden sind, diese nur in einem kürzeren Format vorliegen, liegt es sehr nahe, ein Programm zu entwerfen, das aus den FD-Files die entsprechenden .bmaps generiert.

Auf der Extras-Diskette ist zwar schon ein solches Programm vorhanden, doch erfüllt dieses nicht unbedingt unseren Zweck. Deshalb habe ich ein neues geschrieben, das außerdem noch besondere Möglichkeiten zur Zusammenstellung von .bmaps bietet. Sehen Sie dazu zuerst das Listing:

```
"*********************************
"*                               *
"* Konvertieren von FD zu bmap   *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 13. September "88    *
"* Grueße : Till & Helge         *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************

WINDOW 1
WIDTH 30
Basis% = -30

INPUT "Bitte geben Sie den Library-Namen ein"; N$
FDs$ = "Extras 1.3:FD1.3/"
BMs$ = "Extras 1.3:FD1.3/"

OPEN FDs$+N$+"_lib.fd" FOR INPUT AS 1
OPEN BMs$+N$+".bmap" FOR OUTPUT AS 2
  WHILE EOF(1)=0
```

```
    c$ = INPUT$(1,1)

    If c$ = "*" OR c$ = "#" THEN
      CALL EndRead(CHR$(10))
    Else
      CALL WriteFunktion(c$)
    END If

  WEND
CLOSE 1
CLOSE 2

"--------------------------------------
SUB EndRead(Ende$) STATIC
  c$ = ""

  WHILE c$<>Ende$
    c$ = INPUT$(1,1)
  WEND
END SUB

SUB WriteFunktion(c$) STATIC

 SHARED Basis%
 Funkt$ = "" : A$ = ""

  WHILE c$<>"("
    Funkt$ = Funkt$+c$
    PRINT c$;
    c$ = INPUT$(1,1)
  WEND

  c$ = INPUT$(1,1)
  If c$ <> ")" THEN
    CALL EndRead("(")
    c$ = INPUT$(1,1)
  END If
  Funkt$ = Funkt$ + CHR$(0) + CHR$(PEEK(VARPTR(Basis%)))
  Funkt$ = Funkt$ + CHR$(PEEK(VARPTR(Basis%)+1))
  Basis% = Basis% - 6

  WHILE c$<>")"
    If c$ <> "(" AND c$ <> "/" AND c$ <> "," THEN
      If c$ = "A" THEN
        Funkt$ = Funkt$ + CHR$(VAL(INPUT$(1,1))+9)
      Else
        Funkt$ = Funkt$ + CHR$(VAL(INPUT$(1,1))+1)
```

```
      END If
    END If

    c$ = INPUT$(1,1)
  WEND

  CALL EndRead(CHR$(10))
  WHILE A$<>" " AND A$<>CHR$(13)
    A$ = INKEY$
  WEND
  If A$ = " " THEN
     PRINT#2,Funkt$ + CHR$(0);
     PRINT "*";
  END If

  PRINT

END SUB
```

Listing 5.1: FD-Konverter

Programmbeschreibung

Nach der Eingabe des Library-Namens wird das FD-File aus dem Verzeichnis der Extras-Diskette gelesen. Jede Zeile, die mit einem "#" oder "*" beginnt, wird dezent überlesen, weil sie entweder interne Anweisungen oder aber einen Kommentar enthält.

Für Library wird ein .bmap-File mit gleichem Namen geöffnet, in das jeder Funktions-Name geschrieben wird. Die Parameter-Namen werden auch hier wieder überlesen, weil sie nicht gebraucht werden.

Dafür werden die Register aber um so mehr beachtet und kodiert. Dafür verwendet man untenstehende Tabelle.

Die Adreßregister werden nicht voll ausgelastet, weil das AmigaBASIC ab A5 nicht mehr darauf zugreifen kann. Neben diesen Zahlen für die Register zählt das Programm noch die Nummern der Funktionen mit, um so den Offset für die Library-Tabelle errechnen zu können. Dieser wird ab der Position 30 negativ von der Basisadresse bestimmt. Auch dieser muß in das File geschrieben werden.

Register-Name	Kodierungs-Nummer
D0	1
D1	2
D2	3
D3	4
D4	5
D5	6
D6	7
D7	8
A0	9
A1	10
A2	11
A3	12
A4	13

Tab.: Die Register-Kodierung

Damit das Programm neben seiner Kodier-Arbeit noch etwas Besonderes bietet, wurde eine Abfrage eingebaut, die es ermöglicht, in die .bmap-Datei nicht alle Funktionen zu übernehmen. So spart man Speicherplatz!

Für unsere weitere Arbeit ist es wichtig, daß Sie aus den bestehenden FD-Files Ihrer Extras-Diskette gleich die entsprechenden .bmap-Files kreieren.

Dazu tippen Sie dieses Beispielprogramm ab und wenden es auf die wichtigen Libraries an. Hier ist eine Aufstellung der im weiteren Text benötigten Libraries. Besser wäre es natürlich, wenn Sie gleich alle FD-Files konvertieren, damit Sie später keine Arbeit mehr haben.

```
Exec.Library
Intuition.Library
Graphics.Library
Dos.Library
```

Da unser Programm die .bmap-Files auch auf der Extras-Diskette ablegt, müssen wir diese erst auf die richtige Diskette kopieren. Entweder legt man die Files im LIBS-Verzeichnis der Workbench-Diskette ab oder aber in dem Verzeichnis, in dem das Programm liegt, das die Libraries nutzt. Beide Methoden halte ich für nicht sehr günstig. Denn die Workbench-Diskette ist meistens voll ausgelastet und so nicht mehr für unsere Files aufnahmefähig. Und das

Verzeichnis, in dem unsere Programme liegen, wechselt ständig, wenn man seine Programme ordnet und in verschiedene Verzeichnisse ablegt.

Deshalb ist die beste Methode folgende: Sie haben für das Programmieren im AmigaBASIC ja sowieso eine Arbeits-Diskette angelegt, auf der sich im Hauptverzeichnis das AmigaBASIC befindet. In diesem Hauptverzeichnis legen wir jetzt ein Unterverzeichnis mit dem Namen :bmaps/ an, in das unsere Files gepackt werden können. Im Programm ruft man die Libraries unter dieser Verzeichnis-Bezeichnung auf. Praktisch ist diese Organisation auch, wenn man eine neue Arbeits-Diskette anlegt. Dann muß nur noch das Verzeichnis über die Workbench auf die neue Diskette gelegt werden, und schon läuft der Kopiervorgang.

Sie wissen jetzt, daß das Amiga-Betriebssystem in Libraries organisiert ist und welche Probleme bei der Ansteuerung von AmigaBASIC aus schon zu Anfang auftreten. Mit Hilfe des Beispielprogramms sollte es nun aber keine Schwierigkeit mehr sein, das benötigte .bmap-File zu erstellen, mit dem die Funktionen einer Library von BASIC aus aufgerufen werden können. Am Anfang des Programms muß nur mit

```
LIBRARY ":bmaps/exec.library"
```

eine beliebige oder spezielle Library aktiviert werden. Ab jetzt ist sie in das BASIC eingebunden und kann genutzt werden. Jede Funktion der Library läßt sich wie eine BASIC-Funktion (ein SUB-Programm) nutzen. Parameter können übergeben und sogar empfangen werden!

Es ist nur wichtig, daß man darauf achtet, alle Libraries am Ende des Programms mit

```
CLOSE LIBRARY
```

zu schließen, um den dafür reservierten Speicher freizugeben.

5.1.2 Arbeiten mit den Amiga-Libraries

Nun wissen wir alles über die Library, und deshalb sollen Sie auch endlich in deren Programmierung eingeführt werden. Dazu beginnen wir als erstes mit

einem ganz einfachen Beispiel, um das prinzipielle Vorgehen zu erläutern. Später steigern wir uns dann in dem Schwierigkeitsgrad.

Die Intuition.Library

Das erste Beispielprogramm nutzt eine Funktion der Intuition.Library, die ohne jegliche Parameter, d.h. Werte, aufgerufen werden kann. Und zwar rufen wir DisplayBeep() auf, womit ein Bildschirm-Blinken erreicht wird. Sehen Sie sich dazu das Programm an:

```
"*********************************
"*                               *
"* Test der DisplayBeep()-Funktion *
"* ------------------------------ *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 14. September "88    *
"* Grueße : Kruegge              *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
LIBRARY "intuition.library"
DisplayBeep 0&
LIBRARY CLOSE
```

Listing 5.2: Funktionsaufruf ohne Parameter

Programmbeschreibung

Zwischen dem Öffnen und Schließen der Library wird nicht viel gemacht. Eigentlich wird nur die Funktion aufgerufen. Diese aber läßt alle Screens, ganz besonders natürlich die Workbench-Screen, aufblinken. Den gleichen Effekt können Sie übrigens erzielen, wenn Sie beim AmigaBASIC-Editor versuchen, mit dem Cursor außerhalb des Textes zu laufen.

Eigentlich habe ich ja geschummelt. Die Funktion DisplayBeep() bekommt natürlich doch einen Wert übergeben. Sie erhält den Wert 0, der sie anweist,

alle Screens aufblinken zu lassen. Es gibt leider keine geeignete Funktion, die Wirkung zeigt und gleichzeitig ohne Parameter aufgerufen werden kann.

Die meisten Funktionen benötigen einige Werte, die Einstellungen vornehmen oder einfach Speicheradressen angeben, an denen die Funktion wichtige Informationen findet. Sehen wir uns deshalb noch einige weitere Funktionen der Intuition.Library an, denen Werte übergeben werden müssen. Diese Werte werden im allgemeinen dann hinter der Funktion selbst übergeben. Alle Werte sollten im Format der langen Ganzzahlvariablen übergeben werden, weil das Betriebssystem nur so korrekt arbeitet. Man kennzeichnet die Konstanten deshalb meist mit dem Und-Zeichen ("&").

Gehen wir gleich zum nächsten Beispiel-Programm über, das schon mit einem bestimmten Parameter arbeitet. Die Intuition-Funktion WindowToBack() braucht für eine anstandslose Arbeit nämlich die Speicher-Adresse der Window-Daten. Diese ermitteln wir über den BASIC-Befehl Window().

```
"*********************************
"*                               *
"* Test der Window()-Funktion    *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 14. September "88    *
"* Grueße : Frank Kueppers       *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
LIBRARY "intuition.library"
WindowToBack WINDOW(7)
LIBRARY CLOSE
```

Listing 5.3: WindowToBack() mit Window-Adresse

Programmbeschreibung

Die BASIC-Funktion Window() liefert unter der Angabe 7 die Speicherstelle, an der die Daten für das aktive Window (das Ausgabe-Fenster des AmigaBASIC) gespeichert sind. Diese Adresse, einen Langwert, übergeben wir der Funktion WindowToBack(), die dann dieses Window in den Hintergrund der

Screen legt. Für einen eindeutigen Test empfiehlt es sich, noch einige Verzeichnis-Fenster der Workbench zu öffnen, damit man deutlich die korrekte Reaktion sehen kann.

Als nächstes wollen wir es wieder etwas komplizierter werden lassen. Das dritte Demonstrations-Programm zu den Library-Funktionen nutzt auch eine Intuition-Funktion. Diesmal übergeben wir aber gleich drei Werte. Und hier sehen wir auch den ersten Konflikt mit AmigaBASIC, der bei Funktions-Aufrufen entsteht. AmigaBASIC erlaubt es nicht, die Adresse einer Screen mittels der Window()- und PEEKL()-Funktion zu errechnen. Wir müssen diesen Wert in eine LONG-Variable legen und dann erst der Funktion übergeben. Dieser Umweg ist nötig, weil AmigaBASIC sonst mit Type-Mismatch kontert:

```
"***********************************
"*                                 *
"* Test der MoveScreen()-Funktion  *
"* ------------------------------- *
"*                                 *
"* Autor  : Wolf-Gideon Bleek      *
"* Datum  : 14. September "88      *
"* Grueße : Frank Dzubilla         *
"* Version: 1.0                    *
"* Betriebssystem: V1.2 & V1.3     *
"*                                 *
"***********************************
LIBRARY "intuition.library"
Scr& = PEEKL(WINDOW(7)+46)
FOR i = 1 TO 50
  MoveScreen Scr&, 0&, 1&
NEXT i
FOR i = 1 TO 50
  MoveScreen Scr&, 0&, -1&
NEXT i
LIBRARY CLOSE
```

Listing 5.4: MoveScreen() (Parameter über lange Variablen)

Programmbeschreibung

An der Stelle 46 der Window-Daten befindet sich die Adresse der Screen-Daten, das ist die Screen, auf der sich das Window befindet. Diese Adresse wird in der Variablen Scr& abgelegt und später in den Schleifen beim Funkti-

ons-Aufruf wieder benutzt. Außerdem übergeben wir der Funktion noch zwei Konstanten. Das sind in diesem Fall die Delta-Werte der Verschiebung. Da die Screen zuerst heruntergeschoben wird, ist der Abstand als 1 definiert. Beim Hochschieben verwendet man den inversen Wert.

Wie am Anfang versprochen, wollten wir uns im Schwierigkeitsgrad steigern. Das ist ohne weiteres möglich, denn bisher haben wir nur Werte an die Funktion weitergegeben. Jetzt soll aber die Funktion auch Werte an uns zurückgeben. So z.B. Ergebnisse von Berechnungen oder Speicheradressen, an denen wir neue Daten finden können usw. Für die folgenden Beispielprogramme verwenden wir die Exec.Library, die alle Funktionen zur Verwaltung des Betriebssystems und Systems allgemein enthält. Wir greifen daraus das Gebiet der Speicherverwaltung heraus, zu dem es einige interessante Funktionen gibt. Die erste, die wir kennenlernen, besorgt uns einen Speicherblock in der angewählten Größe und des gewünschten Types. Zurück bekommen wir die Adresse des Speicherblocks. Aber sehen Sie sich zuerst dazu das Programm an, in dem einige Neuerungen vorkommen.

```
"*********************************
"* Speicher-Verwaltung über Exec *
"* ----------------------------- *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 14. September "88    *
"* Grueße : Myrna Pepsi          *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*********************************
LIBRARY ":bmaps/exec.library"
DECLARE FUNCTION AllocMem& LIBRARY
Speicher& = 0
Typ&      = 2^16& + 2&

Speicher& = AllocMem&(100&, Typ&)
If Speicher& <> 0&  THEN
  PRINT "Der Speicher wurde zur Verfuegung gestellt!"
  FreeMem Speicher&, 100&
Else
  PRINT "Es gibt keinen Speicher!"
END If

LIBRARY CLOSE
```

Listing 5.5: Speicherverwaltung

Programmbeschreibung

Sobald eine Funktion einen Wert zurückgeben soll, muß man dem BASIC mitteilen, welcher Art dieser Wert ist. Dafür steht die Deklarationszeile Nach dem Öffnen der Library. Sie stellt fest, daß die Funktion aus der Library stammt und einen Langwert liefert.

Wir verwenden im ganzen Programm zwei Variablen: die erste, Speicher&, zum Festhalten der Speicheradresse, die uns die Funktion liefert. Sie besitzt natürlich den gleichen Datentyp wie die Funktion. Die zweite Variable enthält eine Flagge, die anzeigt, daß der geforderte Speicherbereich gelöscht werden soll und vom Typ Chip-RAM ist. Dann erfolgt der Aufruf der Funktion, indem unserer Adreßvariablen das Ergebnis der Funktion zugewiesen wird. Die Parameter sind hier in Klammern zu setzen, weil wir es mit einer Library-Funktion zu tun haben, die Werte zurückliefert.

Als Ergebnis stehen zwei Möglichkeiten zur Verfügung: Entweder ist der Speicher reserviert worden, dann steht in Speicher& die Startadresse. Oder aber es gab keinen Speicher, der den Bedingungen entsprach, dann wird Null zurückgegeben. Diese Unterscheidung wird gleich nach dem Funktions-Aufruf vorgenommen, um Fehlern vorzubeugen. Nur wenn der Speicher besorgt werden könnte, wird dieser nach der Ausgabe einer Meldung wieder freigegeben. Denn sonst würde der Aufruf der FreeMem()-Funktion zu einem Absturz führen. Benötigt man in seinem Programm mehrere Speicherbereiche - und das wird innerhalb dieses Kapitels sehr oft vorkommen - dann hat es sich als sehr komfortabel erwiesen, wenn man eine Unterroutine schreibt, die den Speicher allokiert, und eine, die den Speicher wieder freigibt.

Das folgende Unterprogramm stellt die beiden Routinen vor. Die erste zur Speicherbesorgung verwendet ein ganz besonderes System zur Verkettung mehrerer Speicherbereiche, damit die Freigabe-Routine mit einfachen Mitteln sofort alle bisher im Programm besorgten Speicherbereiche freigeben kann, ohne daß umständliche Tabellen verwaltet werden müssen.

```
"*********************************
"*                               *
"* Subroutinen zur Speicherverw. *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 15. September "88    *
```

```
"* Grueße : Sabine                *
"* Version: 1.0                   *
"* Betriebssystem: V1.2 & V1.3    *
"*                                *
"**********************************
DEFLNG a-z

LIBRARY ":bmaps/exec.library"
DECLARE FUNCTION AllocMem LIBRARY

MList = 0
Typ   = 2^16 + 2
Haupt:
  GetMem Mem1, 10000&, Typ
  GetMem Mem2, 256&, Typ
  UnGetMem

  END


SUB GetMem(Puffer, Groesse, Typ) STATIC
SHARED MList
  Puffer = AllocMem(Groesse + 8&, Typ)

  If Puffer>0 THEN
    POKEL Puffer  , MList
    POKEL Puffer+4, Groesse + 8&
    MList  = Puffer
    Puffer = Puffer + 8
  Else
    ERROR 7
  END If

END SUB

SUB UnGetMem STATIC
SHARED MList
  WHILE MList>0
    Adresse = PEEKL(MList)
    Groesse = PEEKL(MList+4)
    FreeMem MList, Groesse
    MList = Adresse
  WEND

END SUB
```

Listing 5.6: Speicherverwaltung über Subroutinen

Programmbeschreibung

Im Hauptprogramm des hier vorgestellten Listings werden nur die beiden Speicher-Verwaltungs-Routinen aufgerufen. GetMem besorgt Speicher eines geforderten Typs und einer geforderten Größe. Die Adresse, ab der dieser Speicherblock zu finden ist, wird in der Adreßvariablen übergeben.

Betrachtet man die erste Unter-Routine genauer, so stellt man fest, daß immer 8 Bytes mehr besorgt werden, als eigentlich gefordert werden. Dies hat seinen Grund! Die ersten vier Bytes werden zur Verkettung mehrerer Speicherbereiche verwendet. Hier steht immer die Adresse des nächsten Blocks. In den zweiten vier Bytes wird dann noch die Länge (bzw. die Anzahl der Bytes) des Speicherbereichs gespeichert. So ist der Programmierer entlastet, auch darüber noch Buch zu führen.

Wegen dieser Organisation ist es auch für die Freigabe-Routine so einfach, ihre Aufgabe zu erfüllen. Hier wird nämlich einfach die Speicher-Liste abgeklappert und Block für Block freigegeben. Weil auch die Anzahl der Bytes jedes Speicherbereiches vermerkt wurde, ist es kein Problem, FreeMem() alle geforderten Daten zu übergeben.

Dieses Programm stellt ein Musterbeispiel der Speicherverwaltung dar. Es macht es möglich, für jeden Bedarf Speicherblöcke zu besorgen. Da wir diese auch dringend im späteren Verlauf benötigen, ist es empfehlenswert, wenn Sie die beiden Unter-Routinen als ASCII-Programm abspeichern, damit diese nachher mit MERGE ergänzt werden können.

Es geht weiter. Wie versprochen, steigern wir uns wieder einmal. Nach der einfachen Datenübergabe, dem Empfangen von Werten, gehen wir jetzt eine Stufe höher. Bisher können wir maximal (theoretisch überlegt) so viele Werte an die Funktion übergeben, wie uns Register zur Verfügung gestellt werden. Das sind aber nur 13 Stück, und es gibt bestimmt Fälle, bei denen mehr erforderlich ist. Also hat man sich einen anderen Weg überlegt.

Um mehr Daten zu übergeben, als über die Register erlaubt sind, legt man alle Daten in einem Speicherblock ab, in dem diese nach festgelegter Reihenfolge hintereinander stehen. Diesen als Struktur bezeichneten Datenblock können wir schon einzig und allein an seiner Speicheradresse erkennen.

Also reicht es aus, der Funktion die Startadresse zu übergeben. Sie kann dann selbst die Daten auslesen. Den gleichen Weg geht man bei der Rückgabe von Ergebnissen, die mehr als ein Register beanspruchen. Auch hier wird einfach eine Speicheradresse übergeben, an der man eine festgelegte Struktur von Daten findet.

Das nächste Beispiel-Programm macht dies anhand der Intuition-Funktion OpenWindow() deutlich. Anstatt das Window über den BASIC-Befehl zu öffnen, übergeben wir der Library eine Daten-Struktur, die alle Werte für ein Window enthält. Zurück bekommen wir den Zeiger auf die daraus erstellte Struktur. Aber sehen Sie sich zuerst das Listing an:

```
"*********************************
"*                               *
"*  Window über Intuition öffnen *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 15. September "88    *
"* Grueße : Annemarie            *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
DEFLNG a-z

LIBRARY ":bmaps/exec.library"
DECLARE FUNCTION AllocMem LIBRARY
LIBRARY ":bmaps/intuition.library"
DECLARE FUNCTION OpenWindow LIBRARY

MList = 0
Typ   = 2^16 + 2
Haupt:

  GetMem Ws, 48+10&, Typ

  Text$ = "Test"

  POKEW Ws   ,100          " LeftEdge
  POKEW Ws+ 2,50           " TopEdge
  POKEW Ws+ 4,400          " Width
  POKEW Ws+ 6,100          " Height
  POKEW Ws+ 8,65535&       " Detail- BlockPen
  POKEL Ws+10,512&         " IDCMPFlags
  POKEL Ws+14,15+4096&     " Flags
```

```
  POKEL Ws+18,0&          " FirstGadget
  POKEL Ws+26,Ws+48       " Title
  POKEW Ws+46,1           " ScreenType

  FOR i% = 1 TO LEN(Text$)
     POKE Ws+47+i%, ASC(MID$(Text$, i%, 1))
  NEXT

  WinBase = OpenWindow(Ws)
  FOR t = 1 TO 10000 : NEXT t

  CloseWindow(WinBase)

  END


SUB GetMem(Puffer, Groesse, Typ) STATIC
SHARED MList
  Puffer = AllocMem(Groesse + 8&, Typ)

  If Puffer>0 THEN
    POKEL Puffer  , MList
    POKEL Puffer+4, Groesse + 8&
    MList  = Puffer
    Puffer = Puffer + 8
  Else
    ERROR 7
  END If

END SUB
SUB UnGetMem STATIC
SHARED MList
  WHILE MList>0
    Adresse = PEEKL(MList)
    Groesse = PEEKL(MList+4)
    FreeMem MList, Groesse
    MList = Adresse
  WEND

END SUB
```

Listing 5.7: Window öffnen, Werteübergabe mit Strukturen

Programmbeschreibung

Das Programm nutzt zum Aufbauen einer Daten-Struktur die oben beschriebene Speicherverwaltung. Zuerst wird Speicher in der Größe der Struktur besorgt. Dann schreibt das Hauptprogramm die einzelnen Daten an ihre festgelegte Position. Der Name muß auch im Speicher vorliegen. Deshalb trägt die Struktur als Zeiger die Adresse auf ihr Ende ein. Genau hinter der Struktur sind dann noch einige Bytes frei, die für den Namen verwendet werden.

Zum Öffnen eines Windows werden im einzelnen folgende Werte gebraucht:

- Die Position (x/y) in Pixeln der jeweiligen Screen-Auflösung
- Die Ausdehnung (dx/dy) auch in Pixeln der Screen-Auflösung
- IDCMP-Flaggen zur Nachrichten-Übermittlung
- Allgemein Flaggen für Einstellungen
- Zeiger auf im Window vorhandene Gadgets
- Den Zeiger auf den Text für die Titelzeile
- und den Screen-Typ

Zur Position und Ausdehnung ist sicher nichts zu sagen; beide Wertepaare verstehen sich von selbst und sind beim äquivalenten BASIC-Befehl genauso gehalten. Interessant wird es dann bei den sogenannte IDCMP-Flaggen. Diese sind zur Nachrichten-Übermittlung zwischen Window und Programm vorgesehen. Haben wir ein Window mit Gadgets, müssen darüber ausgelöste Nachrichten natürlich an das Programm weitergegeben werden. Dies erledigt der Messageport. Aber nur, wenn die entsprechende Flagge auch im IDCMP-Register gesetzt ist! Mit den Flaggen für die allgemeinen Einstellungen fordern wir wiederum bestimmte System-Gadgets wie das Drag-Bar oder Sizing-Gadget an. Das kennen Sie auch von den über AmigaBASIC unterstützten Windows.

Hier haben Sie zwei Tabellen. Die erste gibt Auskunft über die Werte, die für ein gewünschtes IDCMP-Flag gesetzt werden müssen. Auch die zweite Tabelle enthält Flaggen-Werte. Diese sind aber mehr für die Einstellungen am Window gedacht.

IDCMP-Flagge	Hex-Wert	Bemerkung
SIZEVERIFY	&H000000001	Nachricht bei ... Sizing-Gadget
NEWSIZE	&H00000002	Neue Window-Größe
REFRESHWINDOW	&H00000004	Grafik-Inhalt zerstört
ACTIVEWINDOW	&H00040000	Aktvierung des Windows
INACTIVEWINDOW	&H00080000	Desaktivierung des Windows
GADGETDOWN	&H00000020	Niederdrücken eines Gadgets
GADGETUP	&H00000040	Loslassen eines Gadgets
CLOSEWINDOW	&H00000200	Schließen des Windows
REQSET	&H00000080	Aktivierung eines Requesters
REQCLEAR	&H00001000	Löschen eines Requesters
REQVERIFY	&H00000800	vorm Setzen des Requesters
MENUPICK	&H00000100	Auswahl eines Menüpunktes
MENUVERIFY	&H00002000	vor der Auswahl
MOUSEBUTTONS	&H00000008	Niederdrücken eines Mausknopfes
MOUSEMOVE	&H00000010	Bewegungen der Maus
DelTAMOVE	&H00100000	Bewegungen mit relativen Angaben
INTUITICKS	&H00400000	Überschreiten eines Zeitlimits
NEWPREFS	&H00004000	Setzen neuer Grundeinstellungen
DISKINSERTED	&H00008000	Einlegen einer Diskette
DISKREMOVED	&H00010000	Entfernen einer Diskette
RAWKEY	&H00000400	Drücken einer Taste (im "rohen" Format)
VANILLAKEY	&H00200000	Drücken einer Taste (im ASCII-Format)
WBENCHMESSAGE	&H00020000	Nachrichten der Workbench
LONELYMESSAGE	&H80000000	

Flag-Name	Hex-Wert	Bemerkung
WINDOWSIZING	&H00000001	Sizing-Gadget
WINDOWDRAG	&H00000002	Dragging-Bar
WINDOWDEPTH	&H00000004	Depth-Arrangement-Gadgets
WINDOWCLOSE	&H00000008	Close-Gadget%
SIZEBRIGHT	&H00000010	Sizing-Gadget im rechten Window-Rand
SIZEBOTTOM	&H00000020	Sizing-Gadget im unteren Window-Rand
NOCAREREFRESH	&H00020000	Keine Sendung von Refresh-Nachrichten
SIMPLE_REFRESH	&H00000040	Einfacher Refresh
SMART_REFRESH	&H00000000	Zwischenspeichern überdeckter Bereiche
SUPER_BITMAP	&H00000080	Ganze Grafik wird zwischengespeichert
BACKDROP	&H00000100	Das Window liegt im Hintergrund
GIMMEZEROZERO	&H00000400	Ein Window mit extra verwaltetem Rand
BORDERLESS	&H00000800	Ein Window ohne grafischen Rand
ACTIVATE	&H00001000	Das Window ist beim Öffnen aktiviert
REPORTMOUSE	&H00000200	Man bekommt Nachrichten über die Mausbewegungen am IDCMP-Port
RMBTRAP	&H00010000	Der rechte Mausknopf wird wie der linke behandelt

Der Zeiger auf die vielleicht vorhandenen Gadgets in der NewWindow-Struktur soll uns hier nicht interessieren. Wir werden ihn am Ende des Kapitels ausführlich benutzen, wenn wir eine File-Select-Box programmieren.

Der Zeiger auf den Namen hingegen ist sehr einfach zu verstehen, und auch der Screen-Typ erfordert keine großen Überlegungen. Solange wir das Window auf der Workbench-Screen öffnen, bleibt dieser Wert bestehen. Fertig!

Betrachten wir jetzt noch einen sehr interessanten Aspekt der Window-Programmierung. Immerhin haben Sie in den beiden Tabellen ständig den Hinweis darauf gefunden, daß man zwischen Programm und Window Nachrichten übermitteln kann. Wir widmen uns jetzt dieser Technik.

Dafür wird im Hauptprogramm nicht einfach eine bestimmte Zeitspanne abgewartet, nach der das Window geschlossen werden soll, sondern ständig die Nachrichten-Leitung abgefragt, ob der Benutzer nicht das Window über das Close-Gadget schließen will.

Kommt eine Nachricht an, wird diese daraufhin in einer Unterroutine überprüft.

Bei einem positiven Verlauf des Tests erhält das Hauptprogramm dann den Auftrag, das Fenster zu schließen.

Hier sind die benötigten Programmteile: erstens die Abfrage im Hauptprogramm, die mit einer Exec-Funktion den Datenkanal ausliest, und zweitens die Unterroutine, in der die ankommende Nachricht auf verschiedene Typen hin untersucht wird.

```
"*********************************
"*                               *
"*  Intuition Message empfangen  *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 16. September "88    *
"* Grueße : Marieanne            *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
```

```
DEFLNG a-z

LIBRARY ":bmaps/exec.library"
DECLARE FUNCTION AllocMem LIBRARY
DECLARE FUNCTION GetMsg LIBRARY
LIBRARY ":bmaps/intuition.library"
DECLARE FUNCTION OpenWindow LIBRARY

MList  = 0
Typ    = 2^16 + 2
CLOSEW = 512&
Lauf   = 1

Haupt:

  GetMem Ws, 48+10&, Typ

  Text$ = "Test"

  POKEW Ws   ,100         " LeftEdge
  POKEW Ws+ 2,50          " TopEdge
  POKEW Ws+ 4,400         " Width
  POKEW Ws+ 6,100         " Height
  POKEW Ws+ 8,65535&      " Detail- BlockPen
  POKEL Ws+10,CLOSEW      " IDCMPFlags
  POKEL Ws+14,15+4096&    " Flags
  POKEL Ws+18,0&          " FirstGadget
  POKEL Ws+26,Ws+48       " Title
  POKEW Ws+46,1           " ScreenType

  FOR i% = 1 TO LEN(Text$)
     POKE Ws+47+i%, ASC(MID$(Text$, i%, 1))
  NEXT

  WinBase  = OpenWindow(Ws)
  UserPort = PEEKL(WinBase+86)

  WHILE Lauf = 1
    IntuiMsg = GetMsg(UserPort)
    If IntuiMsg > 0 THEN GOSUB Message
  WEND


  CloseWindow(WinBase)

  END
```

```
Message:

  MsgTyp = PEEKL(IntuiMsg+20)

  CALL ReplyMsg(IntuiMsg)

  If MsgTyp = CLOSEW THEN
    PRINT "CloseWindow!"
    Lauf = 0
  END If

  Return
```

Listing 5.8: Nachrichten-Übertragung im Window

Programmbeschreibung

Noch einmal, damit Sie nicht beunruhigt sind: Hier wurde nicht das ganze Listing abgedruckt, sondern nur die Teile, an denen Veränderungen vorgenommen wurden.

So sparen wir Arbeit, und Sie bekommen auf der gleichen Anzahl Buchseiten mehr Informationen. Deshalb müssen Sie aber die Routinen aus dem vorhergehenden Programm zu diesen dazukopieren, sonst läuft das Programm nicht!

Das Hauptprogramm wurde in diesem Fall um einiges ergänzt. Aus der Exec.Library benutzen wir eine weitere Funktion, die uns einen Wert zurückliefert.

GetMsg() holt aus der Nachrichtenleitung des Windows eine Nachricht und übergibt uns den Zeiger darauf (wenn eine angekommen ist).

Deshalb wurde zwischen dem OpenWindow()- und dem CloseWindow()-Kommando noch eine besondere Abfrage eingefügt. Nachdem die Variable UserPort definiert wurde, kann über sie in einer Endlos-Schleife der UserPort auf Nachrichten überprüft werden. Liegt eine an, d.h. ein Wert ungleich Null wurde zurückgegeben, springen wir in ein Unterprogramm, das die Nachricht auswertet.

Das Unterprogramm prüft die Nachricht, indem es ihren Typ aus der übergebenen Struktur ausliest. Weil aber der Typ immer genau eine der IDCMP-Flagge ist, die wir gesetzt haben, müssen wir nur auf CLOSEWINDOW testen. Im Falle des Falles wird die Endlos-Schleife aufgebrochen, indem die Lauf-Variable zurückgesetzt wird. Dann schließt das Programm unser Window und gibt den Speicher frei. Fertig!

Über diesen Weg ist es möglich, jede der verschiedenen Nachrichten abzufangen und auszuwerten. Sie müssen dazu nur die entsprechenden IDCMP-Flaggen in der NewWindow-Struktur ergänzen und gleichzeitig die Abfrage auf diese Flaggen ausdehnen.

Unser Ziel, Ihnen zu zeigen, wie man Intuition programmiert, haben wir damit erreicht: Sie können Strukturen mit Werten füllen, diese Daten an Funktionen übergeben und Ergebnisse empfangen. Damit haben Sie das Rüstzeug, um beliebig komplexe Gebilde zu formen. Ein Beispiel dafür bekommen Sie am Ende dieses Kapitels.

Die Dos.Library

Neben den wichtigen Bedienelementen von Intuition gibt es auch noch andere Methoden der Ein- und Ausgabe. Sie werden von der Dos.Library gemanaged, die nicht nur für das Abspeichern und Laden von Disketten zuständig ist. Weil wir es hier mit einem Device-Operation-System zu tun haben, lassen sich mit der Dos.Library selbst die CLI-Windows öffnen, was uns ganz neue Perspektiven zeigt.

Wir wollen uns auf den folgenden Seiten hauptsächlich mit der Verzeichnis-Verwaltung beschäftigen. Damit steuern wir schon auf das Projekt hin, das den Abschluß dieses Kapitels darstellen soll. Eine File-Request-Box, mit der man aus den zur Verfügung stehenden Verzeichnissen das gewünschte File anwählen kann. Aber dazu müssen wir erst einmal ein Verzeichnis laden können.

Das ist auch gar nicht so schwer!

Die Dos.Library bietet mehrere Möglichkeiten, um an die Daten eines Files heranzukommen. Einmal das ganz normale Öffnen, bei dem das primäre Ziel

das Auslesen der Daten ist. Diese Methode ist zwar für die meisten Anwendungen geeignet, aber wir wollen ja keine Daten aus dem File, sondern zu ihm.

Deshalb gibt es noch einen zweiten Befehl, der der Open()-Funktion sehr ähnlich ist. Dieses Kommando heißt Lock(). Es besorgt eine Zugriffs-Struktur zu einem File oder einem Verzeichnis. Über diese ist es dann sehr einfach, Informationen anzufordern. Das erste Beispielprogramm versucht erst einmal nur, den Lock auf ein File zu reservieren:

```
"*********************************
"*                               *
"* File-Zugriff ueber DOS-Lock   *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 11. Juli "88         *
"* Grueße : Peterle              *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
DEFLNG a-z
LIBRARY ":bmaps/dos.library"
DECLARE FUNCTION Lock& LIBRARY

  PRINT "Bitte geben Sie den Namen ein!"
  PRINT
  INPUT "Name:";File$
  PRINT

  File$ = File$ + CHR$(0)
  FileL = Lock(SADD(File$), -2&)

  If FileL <> 0 THEN
    PRINT "Lock erhalten!"
    UnLock(FileL)
  Else
    PRINT "Kein Lock möglich!"
  END If
LIBRARY CLOSE
```

Listing 5.9: Zugriff über Lock reservieren

Programmbeschreibung

In diesem Beispielprogramm nutzen wir die Lock()-Funktion der Dos.Library. Nach der Eingabe des File-Namens (der auch für ein Verzeichnis stehen kann) wird dieser mit Null abgeschlossen und der Funktion übergeben. Diese versucht einen Lesezugriff (-2&) anzumelden. Bei Erfolg findet man in der Variablen FileL einen Wert ungleich Null, der den Zeiger auf die Lock-Struktur darstellt. Am Ende wird dieser dann mit UnLock() wieder freigegeben, um das System freizuhalten.

Mit diesem über den Lock angemeldeten Zugriff wurde uns der Weg zu den File- bzw. Verzeichnis-Informationen freigegeben. Eine weitere Funktion, Examine(), holt alle Daten, die das File-System anbieten kann, in eine leere Struktur.

```
"*********************************
"*                               *
"* File-Informationen mit Examine *
"* ------------------------------ *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 11. September "88    *
"* Grueße : Heinzi               *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
DEFLNG a-z
LIBRARY ":bmaps/dos.library"
DECLARE FUNCTION Lock LIBRARY
DECLARE FUNCTION Examine LIBRARY
LIBRARY ":bmaps/exec.library"
DECLARE FUNCTION AllocMem LIBRARY

MList = 0
Typ   = 2^16
GetMem InfoBlock, 300&, Typ
If InfoBlock > 0 THEN
  PRINT "Bitte geben Sie den Namen ein!"
  PRINT
  INPUT "Name:";File$
  PRINT

  File$ = File$ + CHR$(0)
  FileL = Lock(SADD(File$), -2&)
```

```
  If FileL <> 0 THEN
    PRINT "Lock erhalten!"

    Status = Examine(FileL, InfoBlock)
    PRINT "DiskKey:";PEEKL(InfoBlock)
    PRINT "DirTyp :";PEEKL(InfoBlock+4)
    PRINT "Name   : ";
    i% = 8
    WHILE PEEK(InfoBlock+i%) > 0
      PRINT CHR$(PEEK(InfoBlock+i%));
      i% = i% + 1
    WEND

    UnLock(FileL)
  Else
    PRINT "Kein Lock möglich!"
  END If
END If
UnGetMem
LIBRARY CLOSE
END
```

Listing 5.10: Inhalts-Informationen holen

Programmbeschreibung

Dieses neue Listing ist im Hauptteil um einige Zeilen ergänzt. Die Speicher-Routinen sind allerdings nicht mehr dabei. Sie fügen diese bitte, wie schon mehrfach beschrieben, an das Programm an.

Entscheidend für uns ist die neue Funktion Examine(), die in den InfoBlock, einen Speicherbereich, der am Anfang des Programms reserviert wird, die Daten für einen File- oder Directory-Eintrag holt. Diese lesen wir dann, nachdem der Lock besorgt und die Daten eingetragen wurden, aus dem Speicherbereich aus. Wichtig sind z.B. der Typ (File oder Directory) und der Key (die Nummer des Disketten-Blocks, an dem die eigentlichen Daten liegen). Außerdem wird der Name noch einmal ganz exakt ausgegeben.

Um nun mit dieser Einmal-Ausgabe ein ganzes Verzeichnis auszulesen, fehlt uns nicht mehr viel. Eine neue Funktion und eine Schleife machen das komplett. Die DOS-Funktion ExNext() erledigt dabei den schwierigsten Teil der

Arbeit. Sie erlaubt es nämlich, den nächsten Eintrag aus dem Verzeichnis zu holen, damit wir keinen vergessen. Wir rufen diese Funktion dann im Programm so oft auf, wie Einträge im Verzeichnis sind. Sie meldet sich am Ende mit einer Fehlermeldung.

```
"**********************************
"*                                *
"* Verzeichnis mit ExNext auslesen *
"* ------------------------------ *
"*                                *
"* Autor  : Wolf-Gideon Bleek     *
"* Datum  : 17. September "88     *
"* Grueße : Dieterli              *
"* Version: 1.0                   *
"* Betriebssystem: V1.2 & V1.3    *
"*                                *
"**********************************
DEFLNG a-z
LIBRARY ":bmaps/dos.library"
DECLARE FUNCTION Lock LIBRARY
DECLARE FUNCTION Examine LIBRARY
DECLARE FUNCTION ExNext LIBRARY
LIBRARY ":bmaps/exec.library"
DECLARE FUNCTION AllocMem LIBRARY

MList = 0
Typ   = 2^16
GetMem InfoBlock, 300&, Typ
If InfoBlock > 0 THEN
  PRINT "Bitte geben Sie den Verzeichnis-Namen ein!"
  PRINT
  INPUT "Name:";Dir$
  PRINT

  Dir$ = Dir$ + CHR$(0)
  DirL = Lock(SADD(Dir$), -2&)

  If DirL <> 0 THEN
    Status = Examine(DirL, InfoBlock)
    If PEEKL(InfoBlock+4) <> 2 THEN STOP

    WHILE Status <> 0
      i% = 8
      WHILE PEEK(InfoBlock+i%) > 0
        PRINT CHR$(PEEK(InfoBlock+i%));
        i% = i% + 1
      WEND
```

```
      PRINT
      Status = ExNext(DirL, InfoBlock)
    WEND

    UnLock(FileL)
  Else
    PRINT "Kein Lock möglich!"
  END If
END If
UnGetMem
LIBRARY CLOSE
END
```

Listing 5.11: Inhaltsverzeichnis auslesen

Programmbeschreibung

Das Programm wurde so weit ergänzt, daß nicht nach der Ausgabe des ersten Eintrags abgebrochen wird, sondern die Schleife versucht immer wieder, neue Einträge aus dem Directory auszulesen. Ist das nicht mehr möglich, ist der Status also gleich Null, wird der Lock wieder freigegeben, und das Programm bricht ab, indem es den Speicher freigibt und die Libraries schließt.

Beachten Sie auch hier, daß die beiden Subroutinen Get- und UnGetMem ergänzt werden müssen.

Damit haben wir uns die Möglichkeit geschaffen, aus dem Programm heraus ein beliebiges Verzeichnis irgendeiner Diskette anzuzeigen und vielleicht auszuwerten. Dies werden wir ganz besonders gut ausnutzen können, wenn es am Ende dieses Kapitels darum geht, eine File-Request-Box in BASIC zu programmieren.

Doch bei der ganzen Arbeit mit den Verzeichnissen haben wir eigentlich das zentrale Aufgabengebiet der Dos.Library aus dem Auge verloren. Sie wurde nämlich primär für die Ein- und Ausgabe entwickelt. Deshalb werden wir dieses Gebiet jetzt näher beleuchten. Es ist nämlich über die DOS-Funktionen viel schneller möglich, Daten von der Diskette einzulesen, als dies über AmigaBASIC realisiert wurde. Wir haben damit quasi einen eigenen Fast-Load in der Hand.

Doch dafür müssen wir zuerst die Funktionen kennenlernen. Aber gleich hier zu Anfang tritt schon ein Problem auf. Die Funktionen aus der Dos.Library tragen nämlich die gleichen Namen wie die entsprechenden BASIC-Befehle. Beide heißen Open und Close. Deshalb müssen Sie, wenn Sie mit dem selbstgeschriebenen Konverter arbeiten, zuerst das .fd-File korrigieren, indem Sie vor diese Befehle jeweils ein "x" im Namen setzen, so daß wir xOpen und xClose bekommen. Nun ist es kein Problem mehr, die Funktionen im Programm aufzurufen. Unser erstes Programm soll nichts anderes machen, als nach dem File-Namen zu fragen und das File zu öffnen. Dann wird es gleich wieder geschlossen. Wir setzen später dazwischen eine Lese-Routine, die ihren Geschwindigkeitsvorteil beweisen wird.

```
"*********************************
"*                               *
"* File-Zugriff ueber DOS-Handle *
"* ----------------------------- *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 13. Oktober "88      *
"* Gruesse : Ave                 *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
DEFLNG a-z
LIBRARY ":bmaps/dos.library"
DECLARE FUNCTION xOpen LIBRARY

PRINT "Bitte geben Sie den Namen ein!"
PRINT
INPUT "Name:";File$
PRINT

File$ = File$ + CHR$(0)
FileH = xOpen(SADD(File$), 1005)

If FileH <> 0 THEN
  PRINT "Wir haben das File-Handle erhalten!"
  xClose(FileH)
Else
  PRINT "Kein Zugriff möglich!"
END If

LIBRARY CLOSE
```

Listing 5.12: Zugriff über ein DOS-Handle anmelden

Programmbeschreibung

Das Programm versucht, genau wie das Beispiel mit den Lock-Funktionen, einen Zugriff auf das File zu bekommen. Allerdings lassen sich Handles nur auf Files einrichten. Es ist nicht gestattet - wie es bei den Locks möglich war - ein Handle auf ein Verzeichnis oder Device weisen zu lassen. Das liegt aber auch im unterschiedlichen Zweck der beiden Zugriffsarten.

Während die Locks mehr zur Bearbeitung der Directory-Einträge geeignet waren und die Verwaltung der Verzeichnisse unterstützten, sind Handles zum Lesen und Schreiben von Daten in Files geschaffen. Sie gehen also nicht so tief in die Organisation des Systems ein. Das ist aber auch gar nicht nötig. Mit den mächtigen Lese- und Schreib-Befehlen können wir sehr gut auskommen.

Das nächste Programmbeispiel zeigt die Anwendung der Lese- und der Schreibe-Funktion aus der DOS.Library. Weil auch hier beide Funktionen mit Befehlen des AmigaBASIC kollidieren, wurde wieder jeweils "x" davor gesetzt. Dies sehen Sie an der xOpen()-, xRead()-, xWrite()- und xClose()-Funktion, die aus der DOS.Library stammen.

```
"*********************************
"*                               *
"* Daten sichern mit Write()     *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 24. Oktober "88      *
"* Grueße : Peter Otte           *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
OPTION BASE 1
LIBRARY ":bmaps/dos.library"
DECLARE FUNCTION xOpen& LIBRARY
DECLARE FUNCTION xRead& LIBRARY
DECLARE FUNCTION xWrite& LIBRARY
"               xClose
DIM DatenFeld%(10,10)
" Das DatenFeld mit Zufallszahlen füllen
FOR i = 1 TO 10
  FOR j = 1 TO 10
     DatenFeld%(i,j) = RND*32768
     PRINT DatenFeld%(i,j);" ";
```

```
  NEXT j
  PRINT
NEXT i
PRINT
" Die Daten im File mit gleichen Namen abspeichern
File$ = "DatenFeld"+CHR$(0)
Handle& = xOpen&(SADD(File$), 1006)
  Anzahl& = xWrite&(Handle&, VARPTR(DatenFeld%(1,1)),
200&)
CALL xClose(Handle&)
FOR i = 1 TO 10
  FOR j = 1 TO 10
     DatenFeld%(i,j) = 0
  NEXT j
NEXT i
" Die Daten nach dem Löschen des Feldes wieder einlesen
Handle& = xOpen&(SADD(File$), 1005)
  Anzahl& = xRead&(Handle&, VARPTR(DatenFeld%(1,1)), 200&)
CALL xClose(Handle&)
" Die Daten ausgeben
FOR i = 1 TO 10
  FOR j = 1 TO 10
     PRINT DatenFeld%(i,j);" ";
  NEXT j
  PRINT
NEXT i
PRINT
```

Listing 5.13: Die Lese- und Schreibe-Befehle des DOS

Programmbeschreibung

Zur Demonstration der Geschwindigkeit der Lese- und Schreibe-Befehle der DOS.Library haben wir hier ein Datenfeld mit zufälligen Werten gefüllt und abgespeichert. Dafür wurden aber nicht alle Werte einzeln mit dem PRINT#-Befehl an die Floppy gesendet, sondern das gesamte Datenfeld in einem Rutsch an das File geschickt. Dazu verwenden wir die Schreibe-Funktion des DOS, xWrite(), der zum File-Handle die Startadresse des Speicherbereichs und die Länge des Speicherblocks übergeben wird.

Um die Adresse des Speicherblocks zu ermitteln, in dem die Daten des Feldes stehen, wird die BASIC-Funktion VARPTR() benutzt, die uns die Anfangsadresse zurückliefert. Die Größe eines Speicherblocks läßt sich aus dem Vari-

ablentyp und der Anzahl der Elemente im Feld berechnen. Nach folgender Formel kann man dies ganz allgemein tun, wenn man weiß, wieviel Bytes bestimmte Variablentypen verbrauchen:

```
Größe = 1.Dim x 2.Dim x Bytes
```

wobei der Wert Bytes sich nach der folgenden Tabelle bestimmen läßt:

Variablentyp	Speicherverbrauch
short int	2 Bytes
long int	4 Bytes
normal real	4 Bytes
double real	8 Bytes
strings	5 Bytes + Länge des Strings

Tabelle: Der Speicherverbrauch der Variablen

Beachten Sie besonders bei der Berechnung, daß auch das als Nummer 0 geführte Element eines Feldes zur Dimension dazugezählt werden muß, wenn Sie die obige Formel verwenden. Wir haben dies umgangen, indem wir dieses Feldelement mit OPTION BASE ausgeschaltet haben.

Am Ende des Programms wird das so abgespeicherte Variablenfeld wieder eingelesen. Zuvor löschen wir noch die im Speicher befindlichen Daten, um auf Nummer Sicher zu gehen, daß vielleicht gar nichts geladen wurde. Zum Laden wurde, wie aus dem Programm ersichtlich, die Funtion xRead() verwendet, die genau wie auch xWrite() die gleichen Parameter erfordert. Allerdings liest sie die Daten nicht aus dem Speicher, sondern aus dem Disketten-File und schreibt diese nicht auf Diskette, sondern in den angegebenen Speicherbereich.

Die Graphics.Library

Wie Sie gesehen haben, lassen sich mit der Dos.Library hauptsächlich die Ein- und Ausgabe mit dem Diskettenlaufwerk erledigen. Natürlich greifen die Funktionen auch noch bei Festplatten oder anderen Datenträgern. Wir werden uns jetzt weiterhin mit der Ausgabe beschäftigen. Dies verstehen wir aber

jetzt wortwörtlich; es geht um die grafische Ausgabe auf dem Bildschirm oder einem anderen Sichtgerät.

Es war zwar möglich, die Ausgabe über Intuition zu erreichen, aber hier waren die Möglichkeiten sehr begrenzt. Die Graphics.Library schränkt uns nun überhaupt nicht ein. Es ist eher so, daß wir vor der Fülle der Funktionen gar nicht wissen, was wir zuerst machen können. Es sei deshalb an dieser Stelle auf das Supergrafik-Buch von DATA BECKER verwiesen, das alle Funktionen ausführlich erklärt. Wir werden uns hier wegen Platzmangel auf einige der für unsere Probleme nützlichsten beschränken. Bitte haben Sie dafür Verständnis.

Ein für den Menschen sehr wichtiges Gebiet der Grafikausgabe ist der Text. Es ist die vielfältigste und differenzierteste Ausdrucksweise der heutigen Zeit. Leider ist es im AmigaBASIC auch die langsamste Ausgabe der Welt. Man könnte zwar über einige Warteschleifen die Ausgabe noch weiter verlangsamen, aber ich glaube, daran haben Sie kein Interesse.

Wie wundersam ist es da, wenn man weiß, daß die Graphics.Library eine Funktion anbietet, die um einiges schneller ist als der PRINT-Befehl aus dem AmigaBASIC. Da fragt man sich doch, woran das liegen mag. Aber wir haben keine Zeit für Spekulationen. Wir handeln! Mit dem nachstehenden Listing demónstrieren wir Ihnen, wie schnell und wie einfach die Textausgabe über die Graphics.Library realisiert werden kann. Sehen und staunen Sie!

```
"*********************************
"*                               *
"* Schnelle Textausgabe mit Lib ! *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 01. Oktober "88      *
"* Grueße : RUSHWARE             *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
LIBRARY ":bmaps/graphics.library"

RESTORE
t1a = TIMER
FOR i = 1 TO 5
  READ T$, l&
```

```
  PRINT T$
NEXT i
t1b = TIMER
RESTORE
RastPort& = WINDOW(8)
t2a = TIMER
FOR i& = 1 TO 5
  READ T$, l&
  Move RastPort&, 0&, 50&+i&*8&
  Text RastPort&, SADD(T$), l&
NEXT i&
t2b = TIMER
PRINT : PRINT
PRINT "Erste Zeit :";t1b-t1a
PRINT "Zweite Zeit:";t2b-t2a

DATA "Das ist ein Beispieltext,",25
DATA "der die Geschwindigkeit",23
DATA "der Library-Funktion Text()",27
DATA "demonstrieren soll!",19
DATA "Sehen Sie!!!",12
```

Listing 5.13: Textausgabe über die Graphics.Library

Programmbeschreibung

In den am Ende stehenden DATA-Zeilen sind fünf Texte mit ihren Längen festgehalten. Diese werden in der ersten Schleife ausgelesen und mit PRINT-Befehlen ausgegeben. Parallel dazu wird vom Programm die Zeit gestoppt.

Dann werden die Voraussetzungen für die zweite Schleife geschaffen: der Zeiger auf die DATA-Zeilen wird zurückgesetzt, und ein Zeiger auf den RastPort des Ausgabe-Windows von AmigaBASIC wird in die Variable RastPort gelegt. Dieser wird unbedingt von den Grafik-Funktionen der Graphics.Library benötigt, da diese nicht an ein Window gebunden sind, sondern an einen Port, der die Grafik-Fläche definiert.

Die Schleife liest nun genau wie die erste die Daten aus den DATA-Zeilen, dann aber wird die erste Funktion aus der Library aufgerufen. Es handelt sich hier um Move(), mit dem die Position innerhalb des RastPorts für die Ausgabe bestimmt werden kann. Diese Arbeit müssen wir in diesem Fall selbst übernehmen, weil die Ausgabe-Funktion die Position des Cursors nur in hori-

zontaler Richtung verschiebt. Der Text soll aber immer in einer neuen Zeile gedruckt werden.

Nun kommt endlich die Textausgabe selbst. Der Text()-Funktion wird wie Move() auch die Adresse des RastPorts übergeben, dazu noch die Adresse des Textes und die Anzahl der Zeichen, die wir dafür extra in den DATA-Zeilen gespeichert haben. Die Ausgabe erfolgt dann in Windeseile. Es läßt sich mit dem Auge gar nicht registrieren! Entscheidend ist aber die untrügliche Stoppuhr des Computers. Deshalb werden am Ende die Zeiten ausgegeben.

Bitte beachten Sie bei der Auswertung, daß die erste Schleife nur zwei BASIC-Befehle enthält. In der zweiten Schleife hingegen haben wir den READ-Befehl, der auch in der ersten vorhanden ist, und zusätzlich zwei Aufrufe der Library-Funktionen. Wichtig zu erwähnen sind also die größere Anzahl der Befehle (bei der geringen Menge macht sich einer wirklich bemerkbar) und auch der Vorgang, der beim Aufruf von Library-Routinen vonstatten geht. Es werden nämlich zuerst die Werte in die Maschinenspracheregister geladen und dann erst die Funktionen aufgerufen. Und trotz dieses etwas umständlichen Vorgehens müssen Sie zugeben, daß die zweite Art, Texte auszugeben, ein bißchen schneller ist.

Dies Ausgabe kann jederzeit noch erweitert werden. So besteht die Möglichkeit, einen neuen Zeichensatz zu laden, der mit Text() verwendet wird. Weil das überhaupt keine Schwierigkeit darstellt, wollen wir das gleich machen:

```
"*********************************
"*                               *
"* Verschiedene Zeichensätze     *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 12. Oktober "88      *
"* Grueße : Michaela             *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
DEFLNG a-z
CHDIR ":bmaps"
LIBRARY "graphics.library"
LIBRARY "diskfont.library"
DECLARE FUNCTION OpenFont LIBRARY
DECLARE FUNCTION OpenDiskFont LIBRARY
```

```
INPUT "Zeichensatz: "; Font$
INPUT "Größe:       "; Punkt%
Font$ = Font$ + ".font" + CHR$(0)
TextAttr(0) = SADD(Font$)
TextAttr(1) = 2^16*Punkt%
AlterFont = PEEKL(WINDOW(8)+52)
NeuerFont = OpenFont(VARPTR(TextAttr(0)))
If NeuerFont = 0 THEN
  NeuerFont = OpenDiskFont(VARPTR(TextAttr(0)))
END If
If NeuerFont <> 0 THEN
  SetFont WINDOW(8), NeuerFont
END If
Move WINDOW(8), 0&, 50&
Text WINDOW(8), SADD("Das ist ein Text"), 16&
SetFont WINDOW(8), AlterFont
CloseFont NeuerFont
```

Listing 5.14: Textausgabe mit verschiedenen Zeichensätzen

Programmbeschreibung

Zusätzlich zur Graphics.Library wird die Diskfont.Library geöffnet, die für die Verwaltung der Zeichensätze zuständig ist, die nicht im Speicher des Amiga Platz fanden. Das sind eigentlich fast alle Zeichensätze außer Topaz 8 und 9. Wir benutzen jetzt zwei neue Betriebssystem-Funktionen. OpenFont öffnet einen Zeichensatz, der innerhalb des Speichers verfügbar ist. Das kann z.B. ein schon geladener sein. OpenDiskFont hingegen lädt einen Zeichensatz neu von der Diskette nach, wie der Name schon sagt.

Im Programm geben Sie zuerst den Namen und seine Punktgröße an. Daraufhin startet die Arbeit. Das Programm ergänzt den Namen mit dem Kürzel ".font" und dem Null-Byte. Außerdem werden die Werte in einer Feldvariablen abgelegt. Das erspart das Initialisieren einer Struktur. Nun wird nur noch der alte Zeichensatz zwischengespeichert, und die Wertetabelle kann an die OpenFont()-Funktion übergeben werden. Es wird nachgeprüft, ob der Font im Speicher vorhanden war. Nur wenn dies nicht der Fall ist, schauen wir noch mit OpenDiskFont() auf der Workbench-Diskette im FONT:-Verzeichnis nach. Hier sollte aber auf jeden Fall ein positives Ergebnis kommen.

Es ist am Schluß nur noch nötig, diesen geladenen Zeichensatz im Ausgabe-Fenster zu initialisieren, was durch die Funktion SetFont() erledigt wird. Dann kommt die Textausgabe über die Text()-Funktion, die wir ja wegen ihrer Geschwindigkeitsvorteile nutzen. Am Ende setzen wir ordnungshalber den alten Zeichensatz wieder ein und schließen den geladenen. Fertig!

Wie Sie sehen, ist es ganz einfach, einen neuen Zeichensatz mittels der Diskfont.Library zu öffnen. Es ist nun über die gesamten Text-Funktionen der Graphics.Library möglich, den Text in seinem Stil zu beeinflussen. Auch dafür haben wir ein kleines Programm vorbereitet, das nur um wenige Befehle ergänzt werden mußte und schon erstaunliche Ergebnisse erzielt. Sehen Sie selbst:

```
"*********************************
"*                               *
"* Zeichensätze mit Schrifttypen *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 12. Oktober "88      *
"* Grueße : Anette               *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************
DEFLNG a-z
CHDIR ":bmaps"
LIBRARY "graphics.library"
LIBRARY "diskfont.library"
DECLARE FUNCTION OpenFont LIBRARY
DECLARE FUNCTION OpenDiskFont LIBRARY
DECLARE FUNCTION AskSoftStyle% LIBRARY
DECLARE FUNCTION SetSoftStyle% LIBRARY
INPUT "Zeichensatz: "; Font$
INPUT "Größe:       "; Punkt%
INPUT "Stil-Nummer: "; Stil%
Font$ = Font$ + ".font" + CHR$(0)
TextAttr(0) = SADD(Font$)
TextAttr(1) = 2^16*Punkt%
AlterFont = PEEKL(WINDOW(8)+52)
NeuerFont = OpenFont(VARPTR(TextAttr(0)))
If NeuerFont = 0 THEN
  NeuerFont = OpenDiskFont(VARPTR(TextAttr(0)))
END If
```

```
If NeuerFont <> 0 THEN
  SetFont WINDOW(8), NeuerFont
END If
OldStyle% = AskSoftStyle%(WINDOW(8))
NewStyle% = SetSoftStyle%(WINDOW(8), Stil%, OldStyle%)
Move WINDOW(8), 0&, 50&
Text WINDOW(8), SADD("Das ist ein Text"), 16&
SetFont WINDOW(8), AlterFont
CloseFont NeuerFont
```

Listing 5.15: Textausgabe mit neuen Textattributen

Programmbeschreibung

Die Auswahl eines Zeichensatzes ist eigentlich nur Beiwerk. Was zählt, sind die beiden Zeilen am Ende des Programms, die vor der Ausgabe eingefügt wurden. Mit der ersten Funktion fragen wir den aktuellen Stil des Zeichensatzes ab. Wir erhalten einen Wert als Ergebnis.

Diesen Wert und den gewünschten neuen Stil geben wir der zweiten Funktion als Argumente. Diese setzt den Textstil, und fertig ist unsere Einstellung.

Jetzt müssen Sie nur noch wissen, welche Stil-Nummern was bewirken. Dafür gibt es eine ganz einfache und sehr kurze Tabelle:

Nummer	Stil	Bemerkung
0	normal	Textdarstellung ohne Veränderung
1	unterstreichen	
2	fett	Doppeldruck
4	kursiv	der Text wird schräg dargestellt

Tabelle: Der Textstil

Möchten Sie zwei verschiedene Textstile gleichzeitig verwenden, dann ist das ohne weiteres möglich. Sie müssen dazu nur die Zahlenwerte für jeden einzelnen Textstil addieren

5.2 Programmieren in BASIC wie die Großen

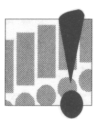
Sie haben im vorhergehenden Teil dieses Kapitels so einiges über das Betriebssystem gelesen. Der Aufbau, die Libraries, der Aufruf und die Programmierung waren Themen, die mit vielen Beispielen belegt wurden. Aber jetzt kommt's!

Damit Sie einen wirklichen Eindruck der AmigaBASIC-Betriebssystem-Programmierung bekommen, haben wir für Sie ein Programm-Modul entwickelt, das aus allen bisher besprochenen Gebieten die einzelnen Teile aufgreift und sie zu einem großen Ganzen zusammenfügt.

Dieses große Programm ist nicht gerade kurz! Wir geben es ja zu! Aber es zeigt, wie man mit Intuition ein Window mit einigen Gadgets aufbaut, wie man (in der Anwendung und nicht in der Theorie) ein Verzeichnis der Diskette einliest und den Anwender daraus einen Namen auswählen läßt; was wir programmiert haben, ist eine File-Request-Box.

Sie können diese in jedem Programm, das die Funktionen Abspeichern und Laden benutzt, verwenden. Bei dieser Vielseitigkeit konnten wir uns einfach nicht bremsen und mußten das Listing für Sie abdrucken.

Natürlich ist das nicht das einzige, was wir als großes Programm anzubieten haben. Das zweite Groß-Listing ist ein Disketten-Editor, mit dem Sie die einzelnen Blöcke Ihrer Diskette unter die Lupe nehmen können. Auch hierfür wenden wir unser Wissen aus dem vorhergehenden Teil des Buches an. Allerdings ist das Programm nicht mit Elementen aus Intuition ausgestattet, weil dies die Länge unnötig erhöht hätte. Sie können es in Eigenarbeit selbst ergänzen und so natürlich auch noch zeigen, daß Sie es verstehen, Ihren Amiga intern zu programmieren.



Bevor es ernst wird, noch ein paar Worte zum Aufbau der folgenden Listings. Weil bisher schon jede Kleinigkeit lang und breit erklärt wurde, haben wir darauf verzichtet, jede Zeile eines Unterprogramms ausführlich zu beschreiben. Am Ende des Listings finden Sie eine Bedienungsanleitung, eine Modulliste und eine Parametererklärung zu allen Unterprogrammen. Damit haben Sie alle Informationen, um sich die Arbeitsweise des Listings zu erarbeiten und gegebenenfalls auch Teile daraus für eigene Programme zu verwenden.

5.2.1 Eine File-Request-Box

Listing:

```
"*********************************
"*                               *
"* File-Request-Box              *
"* ----------------------------- *
"*                               *
"* Autor  : Wolf-Gideon Bleek    *
"* Datum  : 24. September "88    *
"* Grueße : Traute               *
"* Version: 1.0                  *
"* Betriebssystem: V1.2 & V1.3   *
"*                               *
"*********************************

DEFLNG a-z


LIBRARY ":bmaps/exec.library"
DECLARE FUNCTION AllocMem LIBRARY
DECLARE FUNCTION GetMsg LIBRARY

LIBRARY ":bmaps/intuition.library"
DECLARE FUNCTION OpenWindow LIBRARY

LIBRARY ":bmaps/dos.library"
DECLARE FUNCTION Lock LIBRARY
DECLARE FUNCTION Examine LIBRARY
DECLARE FUNCTION ExNext LIBRARY

LIBRARY ":bmaps/graphics.library"


MList = 0
Typ   = 2^16 + 2

Haupt:

  File$ = "Clock" : Pfad$ = "DF0:"
  Titel$ = "File LADEN"

  FileRequest File$, Pfad$, Titel$, Status%

  If Status% = 1 THEN
    PRINT "OK"
```

```
44    Else
45      PRINT "ABBRUCH"
46    END If
47
48    PRINT Pfad$+File$
49
50    END
51
52  "--------------------------------------
53
54  SUB FileRequest (File$, Pfad$, Titel$, Status%) STATIC
55
56  SHARED Typ
57
58  DIM Feld$(100), GadChs(12)
59
60  GADGETDOWN = 32&
61  GADGETUP   = 64&
62
63    Border Rand, -1, -1, 1, 66, 14
64    Border Kasten, 0, 0, 1, 230, 110
65    Border Linie, 0, 10, 1, 220, 0
66
67    IntuiText AbbTxt, 1, 5, 2, "Abbruch", 0&
68    IntuiText OKTxt,  1, 25, 2, "OK", 0&
69    IntuiText InhTxt, 1, 9, 2, "Inhalt", 0&
70    IntuiText FilTxt, 1, -42, 1, "File:", 0&
71    IntuiText DirTxt, 1, -42, 1, "Dir :", 0&
72
73    GetMem Puffer, 8&, Typ
74    GetMem FilePuffer, 256&, Typ
75    GetMem DirPuffer, 256&, Typ
76    GetMem UndoPuffer, 256&, Typ
77
78    FOR i% = 1 TO LEN(File$)
79      POKE FilePuffer+i%-1, ASC(MID$(File$, i%, 1))
80    NEXT i%
81    FOR i% = 1 TO LEN(Pfad$)
82      POKE DirPuffer+i%-1, ASC(MID$(Pfad$, i%, 1))
83    NEXT i%
84
85    PropInfo PropI, 1+4, 0, &HFFFF, 0, 150
86    StringInfo FileI, 255, FilePuffer, UndoPuffer
87    StringInfo DirI, 255, DirPuffer, UndoPuffer
88
89    letztes = 0&
90
91    FOR i% = 1 TO 12
```

```
 92      GadgetDef GadChs(i%), letztes, 32, 7+i%*9, 225, 9, 0, 1,
          1, 0&, 0&, 0&, 10+i%
 93      letztes = GadChs(i%)
 94    NEXT i%
 95
 96    GadgetDef DirGad, letztes, 70, 130, 220, 10, 0, 1, 4,
        Linie, DirTxt, DirI, 6
 97    GadgetDef FilGad, DirGad, 70, 145, 220, 10, 0, 1, 4,
        Linie, FilTxt, FileI, 5
 98    GadgetDef ProGad, FilGad, 270, 15, 20, 110, 0, 1, 3,
        Puffer, 0&, PropI, 4
 99    GadgetDef InhGad, ProGad, 120, 160, 64, 12, 0, 1, 1, Rand,
       InhTxt, 0&, 3
100    GadgetDef AbbGad, InhGad, 210, 160, 64, 12, 0, 1, 1, Rand,
       AbbTxt, 0&, 1
101    GadgetDef OKGad , AbbGad, 30, 160, 64, 12, 0, 1, 1, Rand,
        OKTxt, 0&, 2
102
103    WinDef FileWindow, 120, 20, 310, 180, 32+64+512&,
        2+4+4096&, OKGad, Titel$
104    WinBase = OpenWindow(FileWindow)
105    If WinBase = 0 THEN ERROR 7
106
107    RastPort = PEEKL(WinBase+50)
108    UserPort = PEEKL(WinBase+86)
109    Lauf% = 1
110
111    DrawBorder RastPort, Kasten, 30&, 15&
112
113    WHILE Lauf% = 1
114
115      IntuiMsg = GetMsg(UserPort)
116      If IntuiMsg > 0 THEN
117
118        MsgTyp = PEEKL(IntuiMsg+20)
119        Item   = PEEKL(IntuiMsg+28)
120        GadgetNr% = PEEK(Item+39)
121        CALL ReplyMsg(IntuiMsg)
122
123        If (MsgTyp = GADGETUP) THEN
124          "PRINT "Gadget";GadgetNr%;"up"
125
126          If GadgetNr% = 1 THEN     "Abbruch
127            Lauf% = 0
128            Status% = 0
129          ELSEIF GadgetNr% = 2 THEN "OK
130            Lauf% = 0
131            Pfad$ = ""
```

```
132           File$ = ""
133           bpos = DirPuffer
134           WHILE PEEK(bpos) <> 0
135             Pfad$ = Pfad$ + CHR$(PEEK(bpos))
136             bpos = bpos + 1
137           WEND
138           If RIGHT$(Pfad$,1) <> ":" THEN Pfad$ = Pfad$ + "/"
139           bpos = FilePuffer
140           WHILE PEEK(bpos) <> 0
141             File$ = File$ + CHR$(PEEK(bpos))
142             bpos = bpos + 1
143           WEND
144           Status% = 1
145         END If
146
147         If GadgetNr% = 3 OR GadgetNr% = 6 THEN
148           "PRINT "Inhalt"
149           OffGadget ProGad, WinBase, 0&
150           OffGadget AbbGad, WinBase, 0&
151           OffGadget InhGad, WinBase, 0&
152           OffGadget OKGad, WinBase, 0&
153           YPos% = 1
154           GetDir DirPuffer, Feld$(), Anzahl%
155           Ausgabe Feld$(), YPos%, RastPort
156           Res = 65535&
157           If Anzahl% > 12 THEN Res = Res * 12 / Anzahl%
158           NewModifyProp ProGad, WinBase, 0&, 1+4,
              0, 0, 0, Res, 1
159           "FOR i% = 1 TO Anzahl%
160           "  PRINT Feld$(i%)
161           "NEXT i%
162           SetAPen RastPort, 0&
163           RectFill RastPort, 30, 160, 275, 175
164           OnGadget ProGad, WinBase, 0&
165           OnGadget InhGad, WinBase, 0&
166           OnGadget AbbGad, WinBase, 0&
167           OnGadget OKGad, WinBase, 0&
168         ELSEIF GadgetNr% = 4 THEN
169           Res = PEEK(PropI+4)*256+PEEK(PropI+5)
170           YPos% = Res / 65535& * (Anzahl%-12) + 1
171           "PRINT "Position:";YPos%
172           Ausgabe Feld$(), YPos%, RastPort
173         END If
174
175         If GadgetNr% > 10 THEN
176           Nr% = GadgetNr% - 10 + YPos% - 1
177           If LEFT$(Feld$(Nr%), 5) = "(dir)" THEN
178             bpos = DirPuffer
```

```
179             Text$ = MID$(Feld$(Nr%), 7)+CHR$(0)
180             WHILE PEEK(bpos) <> 0
181               bpos = bpos + 1
182             WEND
183             If PEEK(bpos-1) <> ASC(":") THEN Text$ = "/"
                + Text$
184           Else
185             bpos = FilePuffer
186             Text$ = Feld$(Nr%)+CHR$(0)
187           END If
188           FOR i% = 1 TO LEN(Text$)
189             POKE bpos+i%-1, ASC(MID$(Text$, i%, 1))
190           NEXT i%
191           RefreshGList FilGad, WinBase, 0&, 2&
192         END If
193
194       END If
195
196     END If
197
198   WEND
199
200   CloseWindow(WinBase)
201   CALL UnGetMem
202
203 END SUB
204
205
206
207 SUB GetMem(Puffer, Groesse, Typ) STATIC
208
209 SHARED MList
210
211   Puffer = AllocMem(Groesse + 8&, Typ)
212
213   If Puffer>0 THEN
214     POKEL Puffer  , MList
215     POKEL Puffer+4, Groesse + 8&
216     MList  = Puffer
217     Puffer = Puffer + 8
218   Else
219     ERROR 7
220   END If
221
222 END SUB
223
224 SUB UnGetMem STATIC
225
```

```
226  SHARED MList
227
228    WHILE MList>0
229      Adresse = PEEKL(MList)
230      Groesse = PEEKL(MList+4)
231      FreeMem MList, Groesse
232      MList = Adresse
233    WEND
234
235  END SUB
236
237
238  SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, gad, t$) STATIC
239
240    SHARED Typ
241
242    Size = 48+LEN(t$)+1
243    GetMem bs, Size, Typ
244
245    POKEW bs   ,x%       " LeftEdge
246    POKEW bs+ 2,y%       " TopEdge
247    POKEW bs+ 4,b%       " Width
248    POKEW bs+ 6,h%       " Height
249    POKEW bs+ 8,65535&   " Detail- BlockPen
250    POKEL bs+10,IDCMP    " IDCMPFlags
251    POKEL bs+14,f        " Flags
252    POKEL bs+18,gad      " FirstGadget
253    POKEL bs+26,bs+48    " Title
254    POKEW bs+46,1        " ScreenType
255
256    FOR i%=1 TO LEN(t$)
257       POKE bs+47+i%,ASC(MID$(t$,i%,1))
258    NEXT
259
260  END SUB
261
262  SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, t%,
     i, txt, si, n%) STATIC
263
264    SHARED Typ
265
266    GetMem bs, 44&, Typ  " Gadget-Struktur Länge
267
268    POKEL bs   ,nx       "*NextGadget
269    POKEW bs+ 4,x%       " LeftEdge
270    POKEW bs+ 6,y%       " TopEdge
271    POKEW bs+ 8,b%       " Width
272    POKEW bs+10,h%       " Height
```

```
273    POKEW bs+12,f%        " Flags
274    POKEW bs+14,a%        " Activation
275    POKEW bs+16,t%        " GadgetType
276    POKEL bs+18,i         " GadgetRender
277    POKEL bs+26,txt       "*GadgetText
278    POKEL bs+34,si        " SpecialInfo
279    POKEW bs+38,n%        " GadgetID
280
281  END SUB
282
283  SUB IntuiText(bs, c1%, x%, y%, t$, nx) STATIC
284
285    SHARED Typ
286
287    Size=20+LEN(t$)+1   " IntuiText-Struktur Länge
       + Textlänge + Nullbyte
288    GetMem bs, Size, Typ
289
290    POKE  bs   ,c1%      " FrontPen
291    POKE  bs+ 2,1        " DrawMode
292    POKEW bs+ 4,x%       " LeftEdge
293    POKEW bs+ 6,y%       " TopEdge
294    POKEL bs+12,bs+20    " IText
295    POKEL bs+16,nx       " NextText
296
297    FOR i%=1 TO LEN(t$)
298       POKE bs+19+i%,ASC(MID$(t$,i%,1))
299    NEXT
300
301  END SUB
302
303  SUB Border(bs, x%, y%, c%, b%, h%) STATIC
304
305    SHARED Typ
306
307    GetMem bs, 48&, Typ  " Border-Struktur Länge
       + Koordinatentabelle
308
309    POKEW bs  ,x%         " LeftEdge
310    POKEW bs+2,y%         " TopEdge
311    POKE  bs+4,c%         " FrontPen
312    POKE  bs+7,8          " Count
313    POKEL bs+8,bs+16      "*XY
314
315    FOR i%=0 TO 1
316       POKEW bs+22+i%*4,h%-1
317       POKEW bs+24+i%*4,b%-1
318       POKEW bs+32+i%*4,1
```

```
319       POKEW bs+38+i%*4,h%-1
320       POKEW bs+40+i%*4,b%-2
321     NEXT
322
323   END SUB
324
325
326   SUB PropInfo(bs, Flags%, HBody%, VBody%, HPotRes%,
      VPotRes%) STATIC
327
328     SHARED Typ
329
330     GetMem bs, 22&, Typ
331
332     POKEW bs   ,Flags%
333     POKEW bs+ 6,HBody%
334     POKEW bs+ 8,VBody%
335     POKEW bs+14,HPotRes%
336     POKEW bs+16,VPotRes%
337
338   END SUB
339
340
341   SUB StringInfo (bs, Anzahl%, Puffer, Undo) STATIC
342
343     SHARED Typ
344
345     GetMem bs, 36&, Typ
346
347     POKEL bs   , Puffer
348     POKEL bs+ 4, Undo
349     POKEW bs+10, Anzahl%
350
351   END SUB
352
353
354   SUB GetDir(Dir, Tabelle$(), Num%) STATIC
355
356     Num% = 0
357     GetMem InfoBlock, 300&, Typ
358
359     DirL = Lock(Dir, -2&)
360
361     If DirL <> 0 THEN
362       Status = Examine(DirL, InfoBlock)
363       If PEEKL(InfoBlock+4) <> 2 THEN STOP
364
365       WHILE Status <>0
```

```
366         Tabelle$(Num%) = ""
367         i% = 8
368         WHILE PEEK(InfoBlock+i%) > 0
369           Tabelle$(Num%) = Tabelle$(Num%) +
                               CHR$(PEEK(InfoBlock+i%))
370           i% = i% + 1
371         WEND
372         If PEEKL(InfoBlock+4)>0 THEN
373           Tabelle$(Num%) = "(dir) " + Tabelle$(Num%)
374         END If
375         Status = ExNext(DirL, InfoBlock)
376         If RIGHT$(Tabelle$(Num%), 5) <> ".info" THEN Num% =
                                                        Num% + 1
377       WEND
378
379       UnLock(FileL)
380       Tabelle$(Num%) = ""
381       Num% = Num% - 1
382
383       FOR i% = 1 TO Num% - 1
384         m% = i%
385         FOR j% = i% + 1 TO Num%
386           If Tabelle$(j%) < Tabelle$(m%) THEN m% = j%
387         NEXT j%
388         SWAP Tabelle$(m%), Tabelle$(i%)
389       NEXT i%
390
391     Else
392       PRINT "Kein Lock möglich!"
393     END If
394
395   END SUB
396
397
398   SUB Ausgabe(Tabelle$(), YPos%, RP) STATIC
399
400     SetAPen RP, 0&
401     RectFill RP, 32, 16, 257, 123
402     SetAPen RP, 1&
403
404     FOR i = 0 TO 11
405       File$ = Tabelle$(YPos%+i)
406       Move RP, 33&, 23&+i*9&
407       If File$ = "" THEN EXIT SUB
408       Text RP, SADD(File$), LEN(File$)
409     NEXT i
410
411   END SUB
```

Bedienungsanleitung

Wir haben es hier mit einem sehr großen Unterprogramm zu tun, das viele zusätzliche Unterroutinen benötigt, um seine Arbeit zu verrichten. Für einen ersten Test sollten Sie das Listing abtippen und auf Diskette speichern.

Starten Sie es keinesfalls während der Abtipp-Phase, wenn noch nicht alle Module vorhanden sind oder wenn sich noch Fehler im Listing befinden könnten. Da vom Programm viele System-Routinen benutzt werden, kann es schnell passieren, daß der Amiga "abschmiert" und ihre Arbeit vernichtet!

Haben Sie das Programm fehlerfrei vor sich, können Sie es starten. Nach einiger Zeit, in der AmigaBASIC die .bmap-Files liest, erscheint ein Window in der Mitte unserer Screen, es ist die File-Request-Box.

Damit Sie genau wissen, wie diese aussehen soll, haben wir hier noch einmal eine Hardcopy des Bildschirms abgedruckt:

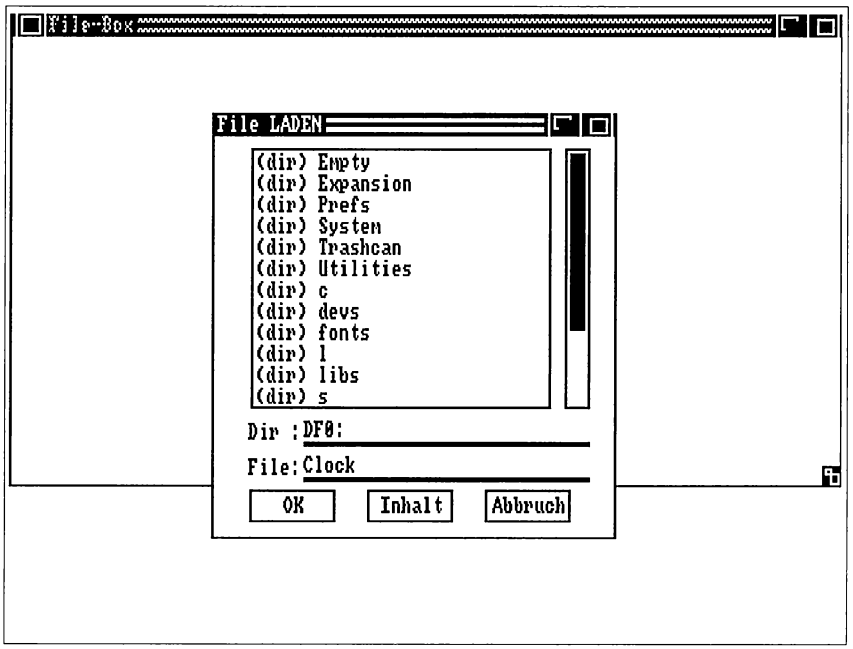


Abb. 5.2: Die File-Request-Box nach dem Start

Die Titelleiste des Windows enthält einen Text, der dem Unterprogramm übergeben wurde. Sie können hier also jederzeit eine Nachricht für den Benutzer unterbringen, die angibt, wofür er einen File-Namen aussuchen soll (z.B. Laden oder Speichern). In dem großen, noch leeren Kasten erscheinen später 12 File-Namen. Rechts daneben ist ein Proportional-Gadget, mit dem bei mehr Einträgen, als in den Kasten passen, ein anderer Bereich der Liste angesehen werden kann. Das Gadget ist momentan nicht verschiebbar, da keine File-Tabelle existiert.

Unter dem Kasten sind zwei String-Gadgets, die für die Eingabe des Pfades und des File-Namens benutzt werden. Hier stehen beim Start die dem Unterprogramm übergebenen Werte, die aber durch Editieren oder Auswahl eines Pfades bzw. Files verändert werden können. In der untersten Reihe des Windows sind noch drei Gadgets zu finden. "OK" bestätigt die Auswahl so, wie sie gerade in den String-Gadgets eingestellt wurde. "Abbruch" beendet das Unterprogramm mit der Nachricht an den Aufrufer, daß es abgebrochen wurde. Die neu eingestellten Pfad- und File-Texte werden nicht zurückgegeben, sondern die Default-Werte, die beim Aufruf übergeben wurden.

Das Gadget "Inhalt" liest den Inhalt des angewählten Verzeichnisses ein. Dabei werden alle Aktions-Gadgets ausgeschaltet, damit nicht zwischendurch ein Gadget unbeabsichtigt angeklickt wird. Nach dem Einlesen des Inhaltsverzeichnisses wird dieses im Kasten ausgegeben, das Proportional-Gadget richtig eingestellt und die Gadgets wieder aktiviert. Jetzt können Sie durch Anklicken eines File- oder Directory-Namens diesen in die String-Gadgets ohne Eingabe der Buchstaben übernehmen.

Modulliste

```
SUB FileRequest (File$, Pfad$, Titel$, Status%) STATIC
```

Funktion

Dieses Hauptmodul stellt das Window auf, in dem sich alle Gadgets, die File-Liste und die Eingabe zusammenfinden werden. Es ruft alle anderen Unterroutinen auf, die kleinere Aufgaben übernehmen. Einen großen Teil dieses

Unterprogramms nimmt die Abfrage der Gadgets in Anspruch. Je nach Auswahl werden wieder andere Subroutinen aufgerufen.

Parameter

File$ Name des Default-File-Namens, hier steht später auch der Rückgabewert drin

Pfad$ Name des Default-Directory-Namens, auch hierin wird der ausgewählte Pfad zurückgegeben

Titel$ ist der Text, der in der Titelleiste des Windows erscheinen soll

Status% wird nur zurückgegeben; man erhält hier einen Pointer, der anzeigt, ob mit "OK" oder "Abbruch" beendet wurde

Speicher-Routinen

```
SUB GetMem(Puffer, Groesse, Typ) STATIC
```

Funktion

Die Subroutine besorgt einen Speicherbereich aus dem System-Pool in der angegebenen Größe und vom angegebenen Typ. Dabei kann der Typ kennzeichnen, ob es sich um Fast- oder Chip-RAM handeln soll und ob dieses gelöscht werden muß.

Parameter

Puffer Rückgabe-Variable, die die Speicheradresse enthält

Groesse gibt die Anzahl der zu reservierenden Bytes an. Der Wert wird immer auf eine durch 8 teilbare Zahl ergänzt.

Typ markiert die Eigenschaften des Speicherbereichs. 2^16 = löschen, 2^0 = jeder Speichertyp, 2^1 = Chip-MEM, 2^2 = Fast-Mem

```
SUB UnGetMem STATIC
```

Funktion

Diese Routine gibt den gesamten im Programm reservierten Speicher wieder frei. Weil alle Bereiche untereinander verknüpft sind, kann nicht nur ein Teil davon dem System zurückgegeben werden. Darauf ist zu achten!

Intuition-Routinen

Diese Routinen erlauben die einfache Arbeit mit der Intuition.Library. Dabei werden die angegebenen Werte in die Strukturen eingetragen und Teile mancher Strukturen mit Standard-Werten vorinitialisiert.

```
SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, gad, t$) STATIC
```

Funktion

Diese Subroutine definiert eine NewWindow-Struktur. Sie trägt alle angegebenen Werte ein und reserviert zusätzlich einen Speicherbereich für den Text, der als Überschrift dargestellt werden soll. Dieser wird dorthin kopiert und der Zeiger in die Struktur eingetragen.

Parameter

bs ist die Basis-Adresse der Struktur, sie wird zurückgeliefert

x% x-Position des Windows

y% y-Position des Windows

b% die Breite des Windows in Pixeln

h% die Höhe des Windows in Pixeln

IDCMP die Flaggen für die Nachrichtenübertragung

f allgemeine Flaggen für das Window; hier werden die geforderten System-Gadgets vermerkt

gad ein Zeiger auf das erste Gadget einer Liste, die im Window dargestellt werden sollen

t$ die Zeichenketten, die in der Titelleiste des Windows erscheinen sollen

```
SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, t%, i, txt, si, n%) STATIC
```

Funktion

GadgetDef arbeitet ähnlich wie WinDef und definiert eine Struktur für ein neues Gadget. Auch dafür wird ein Speicherbereich in der nötigen Größe besorgt und mit den gelieferten Werten gefüllt.

Parameter

bs ist die Basis-Adresse der initialisierten Struktur

nx ein Zeiger auf ein weiteres Gadget (zur Verkettung zu einer Gadget-Liste)

x% x-Position im Window

y% y-Position im Window

b% die Breite des Klickbereiches in Pixeln

h% die Höhe des Klickbereiches in Pixeln

f% Flaggen-Wert zur Einstellung der Eigenschaften des Gadgets

a% Flaggen für den Aktivierungs-Modus

t% gibt den Gadget-Typ an

i Zeiger auf eine eventuelle Image-Struktur

txt Zeiger auf eine eventuelle IntuiText-Struktur, die mit dem Gadget verbunden werden soll

si Zeiger auf eine benötigte SpecialInfo-Struktur, die weitere Informationen zum Gadget enthält

n% Identifikations-Nummer für eigenen Gebrauch

```
SUB IntuiText(bs, c1%, x%, y%, t$, nx) STATIC
```

Funktion

Mit dieser Unterroutine wird eine IntuiText-Struktur mit Daten gefüllt, die man bei Intuition dazu benutzt, Texte auszugeben. Wir verwenden die Struktur ganz speziell für die Texte in den Gadgets, die nur so ausgegeben werden können.

Parameter

bs ist die Basis-Adresse der Struktur

c1% die Zeichenfarbe, mit der der Text gedruckt wird

x% relative x-Position des Textes im Window

y% relative y-Position des Textes im Window

t$ der Text selbst

nx Zeiger auf einen damit zu verkettenden Text

```
SUB Border(bs, x%, y%, c%, b%, h%) STATIC
```

Funktion

Zum Zeichnen von Linien gibt es zwar einen Befehl der Graphics.Library, aber auch die Intuition.Library stellt die Funktion zur Verfügung. Dazu bedient man sich der Border-Struktur, in der die Anzahl der Koodinaten-Punkte vermerkt ist, die verbunden werden sollen. Zusätzlich benötigen wir noch die Tabelle mit den Koordinaten. Alles dies wird von der Unterroutine erledigt: sie besorgt den Speicher für Struktur und Tabelle und füllt beide mit Daten.

Parameter

bs ist die Basis-Adresse der Struktur

x% relative x-Position des Kastens im Window

y% relative y-Position des Kastens im Window

c% Farbe, mit der der Border gezeichnet werden soll

b% die Breite des Kastens in Pixeln

h% die Höhe des Kastens in Pixeln

```
SUB PropInfo(bs, Flags%, HBody%, VBody%, HPotRes%, VPotRes%) STATIC
```

Funktion

Für ein Proportional-Gadget werden mehr Daten benötigt, als in der Gadget-Struktur untergebracht werden können. An dieser Stelle macht sich der Zeiger

SpecialInfo in der Gadget-Struktur bezahlt. Er zeigt auf die nötige Erweiterungs-Struktur, die wir mit diesem Unterprogramm initialisieren.

Parameter

bs ist die Basis-Adresse der Info-Struktur

Flags% Flaggen, die den Betriebsmodus des Gadgets genauer bestimmen

HBody% horizontale Position des Schiebers (im Verhältnis zum gesamten Kasten)

VBody% vertikale Position des Schiebers (im Verhältnis zum gesamten Kasten)

HPotRes% horizontale Auflösung des Schiebers (in Anteilen zum Ganzen)

VPotRes% vertikale Auflösung des Schiebers (in Anteilen zum Ganzen)

```
SUB StringInfo (bs, Anzahl%, Puffer, Undo) STATIC
```

Funktion

Genau wie PropInfo füllt auch StringInfo eine Zusatz-Struktur mit Daten für ein String-Gadget. Die Unterroutine besorgt allerdings nicht den Speicher, der für die Textpuffer benötigt wird. Das muß vom Hauptprogramm aus geschehen.

Parameter

bs ist die Basis-Adresse der Info-Struktur

Anzahl% ist die Anzahl der Zeichen, die eingegeben werden können

Puffer ist ein Zeiger auf den Text-Puffer

Undo ist ein Zeiger auf den Undo-Puffer, in dem der unbearbeitete Text zwischengespeichert wird

Directory-Verwaltung

```
SUB GetDir(Dir, Tabelle$(), Num%) STATIC
```

Funktion

Diese Routine liest das unter Dir angegebene Verzeichnis von der Diskette ein. Dabei werden alle Einträge, die mit einem .info enden, überlesen, weil es sich nur um Files der Workbench handelt. Außerdem sortiert die Routine zum Schluß das Feld Tabelle$(), in dem es die Liste der File-Einträge abgelegt hat, alphabetisch. Vor die Namen der Verzeichnisse wird der Text (dir) gesetzt, um sie von anderen zu unterscheiden.

Parameter

Dir Zeiger (Adresse) auf den Text, in dem der Pfad des Verzeichnisses angegeben ist

Tabelle$() das Feld, in dem das sortierte Verzeichnis abgelegt wird

Num% Anzahl der Einträge dieses Verzeichnisses

Verzeichnis-Ausgabe

```
SUB Ausgabe(Tabelle$(), YPos%, RP) STATIC
```

Funktion

Diese Unter-Routine gibt die Einträge des Feldes Tabelle$() ab dem Element YPos% in dem Window der File-Request-Box aus. Dafür benötigt es den Zeiger auf den RastPort des Windows, der mit RP übergeben wird. Die Ausgabe

erfolgt mit maximal 12 Zeilen und wird abgebrochen, sobald ein Leer-Eintrag im Feld gefunden wird, der das Ende kennzeichnet. Vor der Ausgabe wird der Bereich gelöscht.

Parameter

Tabelle$() das Textfeld, in dem die Verzeichnis-Einträge abgelegt sind

YPos% Nummer des ersten Elements, das ausgegeben werden soll

RP Adresse des RastPorts, in dem die Ausgabe stattfindet

5.2.2 Ein Disk-Monitor

Listing:

```
1  "************************************************
2  "*                                              *
3  "*  Programm:       Disk-Monitor.BAS            *
4  "*                                              *
5  "*  Sprache:        AmigaBASIC                  *
6  "*                                              *
7  "*  Autor:          Mirko Langlotz              *
8  "*                                              *
9  "*  Datum:          28.09.1988                  *
10 "*                                              *
11 "************************************************
12
13 LIBRARY ":bmaps/exec.library"
14 LIBRARY ":bmaps/graphics.library"
15
16 DECLARE FUNCTION AllocMem&   LIBRARY
17 DECLARE FUNCTION FindTask&   LIBRARY
18 DECLARE FUNCTION OpenDevice% LIBRARY
19 DECLARE FUNCTION DoIO%       LIBRARY
20 DECLARE FUNCTION FindName&   LIBRARY
21
22 MD% = 16
23 DIM Feld%(511)
24
```

```
25  Hauptprogramm:
26     CLS
27     COLOR 3
28     PRINT "Disk-Monitor 1.0"
29     COLOR 1
30     PRINT
31     PRINT "<1> Block editieren"
32     PRINT
33     PRINT "<2> File in Boot-Block übertragen"
34     PRINT "<3> Boot-Block in File übertragen"
35     PRINT
36     PRINT "<4> File in Block übertragen"
37     PRINT "<5> Block in File übertragen"
38     PRINT
39     PRINT "<0> Programm verlassen"
40     PRINT
41     COLOR 3
42     PRINT "Ihre Wahl :";
43     COLOR 1
44     I$=""
45     WHILE I$<"0" OR I$>"5"
46        I$=INKEY$
47     WEND
48     ON VAL(I$) GOTO Block, FileBoot-Block, Boot-
       BlockFile, FileBlock, BlockFile
49
50  CLS
51  LIBRARY CLOSE
52  END
53
54  "*****************************************
55  "*                                       *
56  "*  Programmteil:  Block editieren       *
57  "*                                       *
58  "*****************************************
59
60  Block:
61     CLS
62     COLOR 3
63     PRINT "Block bearbeiten"
64     COLOR 1
65     PRINT
66     GOSUB Laufwerk
67     PRINT
68     INPUT "Blocknummer (0-1759) :",I$
69     BlockNummer%=VAL(I$)
70     If BlockNummer%<0 OR BlockNummer%>1759 THEN Block
71     Disk "r",BlockNummer%,Laufw%,Feld%()
```

```
 72     If Fehler$<> "" THEN Fehlerausgabe
 73     Zeile%=0
 74     XPos%=0
 75     YPos%=0
 76
 77     GOSUB Display
 78     GOSUB CRSR
 79
 80     I$=""
 81     WHILE I$<CHR$(129) OR I$>CHR$(130)
 82        I$=INKEY$
 83        If ASC(I$+CHR$(0))=28 THEN GOSUB CUp
 84        If ASC(I$+CHR$(0))=29 THEN GOSUB CDown
 85        If ASC(I$+CHR$(0))=30 THEN GOSUB CRight
 86        If ASC(I$+CHR$(0))=31 THEN GOSUB CLeft
 87        GOSUB Char
 88     WEND
 89
 90     If I$=CHR$(130) THEN Hauptprogramm
 91
 92  SaveBlock:
 93     CLS
 94     COLOR 3
 95     PRINT "Block";BlockNummer%;
            "auf Diskette in DF";HEX$(Laufw%);" schreiben"
 96     COLOR 1
 97     PRINT
 98     If BlockNummer%>1 THEN GOSUB Summe
 99     PRINT "Einverstanden (Ja/";
100     COLOR 3
101     PRINT "N";
102     COLOR 1
103     PRINT "ein) ? ";
104     I$=""
105     WHILE UCASE$(I$)<>"J" AND UCASE$(I$)<>"N"
                                  AND I$<>CHR$(13)
106        I$=INKEY$
107     WEND
108     If UCASE$(I$)="J" THEN
109        PRINT "Ja"
110        Disk "w",BlockNummer%,Laufw%,Feld%()
111        If Fehler$<>"" THEN Fehlerausgabe
112     END If
113  GOTO Hauptprogramm
114
115  Display:
116     CLS
117     GOSUB EditMenu
```

```
118     FOR I%=Zeile% TO Zeile%+MD%-1
119        DZeile%=I%
120        GOSUB DisplayZeile
121     NEXT
122  Return
123
124  EditMenu:
125     GOSUB ClearEditMenuBereich
126     LOCATE 1,1
127     COLOR 3
128     PRINT "Block";BlockNummer%;"    ";
                            "Zugriff auf DF"+HEX$(Laufw%)
129     COLOR 1
130     PRINT
131     PRINT "<F1> Speichern / <F2> Hauptmenü /
               <F3> 000 / <F4> 100 /
               <F5> Hex"
132     PRINT "Cursor-Tasten  / Ascii-Eingabe"
133  Return
134
135  ClearEditMenuBereich:
136     LOCATE 1,1
137     I$=""
138     FOR I%=0 TO 70
139        I$=I$+" "
140     NEXT
141     FOR I%=1 TO 5
142        PRINT I$
143     NEXT
144  Return
145
146  DisplayZeile:
147     LOCATE DZeile%-Zeile%+7,1
148     h$=HEX$(DZeile%*16)
149     WHILE LEN(h$)<3
150        h$="0"+h$
151     WEND
152     PRINT h$;TAB(7);
153     FOR x%=0 TO 15
154        h$=HEX$(Feld%(DZeile%*16+x%))
155        If LEN(h$)=1 THEN h$="0"+h$
156        PRINT h$;" ";
157     NEXT
158     PRINT "  ";
159     FOR x%=0 TO 15
160        If (Feld%(DZeile%*16+x%) AND 127) < 32 THEN
161           COLOR 2
```

```
162           PRINT CHR$(Feld%(DZeile%*16+x%)+64);
163           COLOR 1
164        Else
165           PRINT CHR$(Feld%(DZeile%*16+x%));
166        END If
167     NEXT
168     CRSR.Flag%=0
169  Return
170
171  CRSR:
172     GOSUB CCursor
173     GOSUB RectInv
174     CRSR.Flag%=1
175  Return
176
177  CCursor:
178     If CRSR.Flag%=1 THEN
179        GOSUB RectInv
180        CRSR.Flag%=0
181     END If
182  Return
183
184  RectInv:
185     CALL SetDrMd (WINDOW(8),2)
186     xol%=(56+XPos%)*8
187     yol%=(6+YPos%-Zeile%)*8
188     xur%=xol%+7
189     yur%=yol%+7
190     CALL RectFill (WINDOW(8),xol%,yol%,xur%,yur%)
191     xol%=(6+XPos%*3)*8
192     yol%=(6+YPos%-Zeile%)*8
193     xur%=xol%+15
194     yur%=yol%+7
195     CALL RectFill (WINDOW(8),xol%,yol%,xur%,yur%)
196     CALL SetDrMd (WINDOW(8),1)
197  Return
198
199  CUp:
200     If YPos%>0 THEN
201        GOSUB CCursor
202        If YPos%>Zeile% THEN
203           YPos%=YPos%-1
204        Else
205           If Zeile%>0 THEN
206              GOSUB CCursor
207              Zeile%=Zeile%-1
208              YPos%=YPos%-1
209              SCROLL (0,6*8)-(72*8-1,(6+MD%)*8-1),0,8
```

```
210              DZeile%=Zeile%
211              GOSUB DisplayZeile
212           END If
213        END If
214        GOSUB CRSR
215     END If
216  Return
217
218  CDown:
219     If YPos%<512/16 THEN
220        GOSUB CCursor
221        If YPos%<Zeile%+MD%-1 THEN
222           YPos%=YPos%+1
223        Else
224           If Zeile%<512/16-MD% THEN
225              GOSUB CCursor
226              Zeile%=Zeile%+1
227              YPos%=YPos%+1
228              SCROLL (0,6*8)-(72*8-1,(6+MD%)*8-1),0,-8
229              DZeile%=Zeile%+MD%-1
230              GOSUB DisplayZeile
231           END If
232        END If
233        GOSUB CRSR
234     END If
235  Return
236
237  CRight:
238     GOSUB CCursor
239     If XPos%<15 THEN
240        XPos%=XPos%+1
241     Else
242        If YPos%<512/16-1 THEN
243           XPos%=0
244           GOSUB CDown
245        END If
246     END If
247     GOSUB CRSR
248  Return
249
250  CLeft:
251     GOSUB CCursor
252     If XPos%>0 THEN
253        XPos%=XPos%-1
254     Else
255        If YPos%>0 THEN
256           XPos%=15
257           GOSUB CUp
```

```
258         END If
259      END If
260      GOSUB CRSR
261   Return
262
263   Char:
264      If I$="" THEN Return
265      If ASC(I$)=131 THEN
266         GOSUB CCursor
267         Zeile%=0
268         XPos%=0
269         YPos%=0
270         GOSUB Display
271         GOSUB CRSR
272         Return
273      END If
274      If ASC(I$)=132 THEN
275         GOSUB CCursor
276         Zeile%=16
277         XPos%=0
278         YPos%=16
279         GOSUB Display
280         GOSUB CRSR
281         Return
282      END If
283      If ASC(I$)=133 THEN
284         NeuE:
285         GOSUB ClearEditMenuBereich
286         LOCATE 1,1
287         PRINT "Wert hexadezimal eingeben"
288         PRINT "($";HEX$(XPos%+YPos%*16);"):";
289         INPUT "",I$
290         If LEN(I$)>2 THEN NeuE
291         If LEN(I$)<1 THEN KeineEingabe
292         I$=UCASE$(I$)
293         Hilfs$=LEFT$(I$,1)
294         If (Hilfs$<"0" OR Hilfs$>"9") AND (Hilfs$<"A"
            OR Hilfs$>"F") THEN NeuE
295         Hilfs$=RIGHT$(I$,1)
296         If (Hilfs$<"0" OR Hilfs$>"9") AND (Hilfs$<"A"
            OR Hilfs$>"F") THEN NeuE
297         Feld%(XPos%+YPos%*16)=VAL("&h"+I$)
298         DZeile%=YPos%
299         GOSUB DisplayZeile
300         GOSUB CRight
301         KeineEingabe:
302         GOSUB EditMenu
303         Return
```

```
304     END If
305     If ASC(I$)>=128 AND ASC(I$)<128+32 THEN Return
306     If ASC(I$)>= 28 AND ASC(I$)<    32 THEN Return
307     Feld%(YPos%*16+XPos%)=ASC(I$)
308     DZeile%=YPos%
309     GOSUB DisplayZeile
310     GOSUB CRight
311  Return
312
313  "****************************************
314  "*                                      *
315  "*  Programmteil:  Fehlermeldung        *
316  "*                                      *
317  "****************************************
318
319  Fehlerausgabe:
320     CLS
321     COLOR 3
322     PRINT Fehler$
323     COLOR 1
324     PRINT
325     PRINT "Taste drücken !"
326     I$=""
327     WHILE I$=""
328        I$=INKEY$
329     WEND
330  GOTO Hauptprogramm
331
332  "****************************************
333  "*                                      *
334  "*  Programmteil:  Pruefsumme berechnen *
335  "*                                      *
336  "****************************************
337
338  Summe:
339     ASummeH&=Feld%(20)*256+Feld%(21)
340     ASummeL&=Feld%(22)*256+Feld%(23)
341     Summe$=HEX$(ASummeL&)
342     WHILE LEN(Summe$) < 4
343        Summe$="0"+Summe$
344     WEND
345     Summe$=HEX$(ASummeH&)+Summe$
346     WHILE LEN(Summe$) < 8
347        Summe$="0"+Summe$
348     WEND
349     PRINT "Alte Summe ist ";Summe$
350     PRINT "Neue Summe ist ";
351     NSummeL&=0
```

```
352     NSummeH&=0
353     FOR I%=0 TO 512-4 STEP 4
354        If I%<>20 THEN
355           NSummeL&=NSummeL&+Feld%(I%+2)
              *256+Feld%(I%+3)
356           If NSummeL&>=256*256 THEN
357              NSummeL&=NSummeL&-256*256
358              NSummeH&=NSummeH&+1
359              If NSummeH&>=256*256 THEN
360                 NSummeH&=NSummeH&-256*256
361              END If
362           END If
363           NSummeH&=NSummeH&+Feld%(I%+0)
              *256+Feld%(I%+1)
364           If NSummeH&>=256*256 THEN
365              NSummeH&=NSummeH&-256*256
366           END If
367        END If
368     NEXT
369     Hilfs$=CHR$(INT(NSummeH&/256))
370     Hilfs$=Hilfs$+CHR$(NSummeH&-INT(NSummeH&/256)*256)
371     Hilfs$=Hilfs$+CHR$(INT(NSummeL&/256))
372     Hilfs$=Hilfs$+CHR$(NSummeL&-INT(NSummeL&/256)*256)
373     NSumme&=-CVL(Hilfs$)
374     NSummeH&=PEEK(VARPTR(NSumme&)+0)
        *256+PEEK(VARPTR(NSumme&)+1)
375     NSummeL&=PEEK(VARPTR(NSumme&)+2)
        *256+PEEK(VARPTR(NSumme&)+3)
376     Summe$=HEX$(NSummeL&)
377     WHILE LEN(Summe$) < 4
378        Summe$="0"+Summe$
379     WEND
380     Summe$=HEX$(NSummeH&)+Summe$
381     WHILE LEN(Summe$) < 8
382        Summe$="0"+Summe$
383     WEND
384     PRINT Summe$
385     PRINT
386     PRINT "Neue Summe übernehmen (";
387     COLOR 3
388     PRINT "J";
389     COLOR 1
390     PRINT "a/Nein) ? ";
391     I$=""
392     WHILE UCASE$(I$)<>"J" AND UCASE$(I$)<>"N"
        AND I$<>CHR$(13)
393        I$=INKEY$
394     WEND
```

```
395     If UCASE$(I$)<>"N" THEN
396        PRINT "Ja"
397        PRINT
398        Feld%(20)=INT(NSummeH&/256)
399        Feld%(21)=NSummeH&-INT(NSummeH&/256)*256
400        Feld%(22)=INT(NSummeL&/256)
401        Feld%(23)=NSummeL&-INT(NSummeL&/256)*256
402     Else
403        PRINT "Nein
404        PRINT
405     END If
406  Return
407
408  "****************************************
409  "*                                      *
410  "*  Programmteil:  Laufwerk wählen      *
411  "*                                      *
412  "****************************************
413
414  Laufwerk:
415     PRINT "Welches Laufwerk? DF";
416     I$=""
417     WHILE I$<"0" OR I$>"3"
418        I$=INKEY$
419     WEND
420     PRINT I$
421     Laufw%=VAL (I$)
422  Return
423
424  "****************************************
425  "*                                      *
426  "*  Programmteil:  Datei <> Block       *
427  "*                                      *
428  "****************************************
429
430  FileBoot-Block:
431     CLS
432     COLOR 3
433     PRINT "File in Boot-Block übertragen"
434     COLOR 1
435     PRINT
436     INPUT "FileName:",FileName$
437     OPEN "i",#1,FileName$
438     If LOF(1)<>1024 THEN
439        Fehler$="Filelänge <> 1024"
440        CLOSE 1
441        GOTO Fehlerausgabe
442     END If
```

```
443     In$=INPUT$(1024,1)
444     CLOSE 1
445     PRINT
446     GOSUB Laufwerk
447     PRINT
448     FOR I%=0 TO 511
449        Feld%(I%)=ASC(MID$(In$,I%+1,1))
450     NEXT
451     Disk "w",0,Laufw%,Feld%()
452     If Fehler$<> "" THEN Fehlerausgabe
453     FOR I%=0 TO 511
454        Feld%(I%)=ASC(MID$(In$,I%+1+512,1))
455     NEXT
456     Disk "w",1,Laufw%,Feld%()
457     If Fehler$<> "" THEN Fehlerausgabe
458  GOTO Hauptprogramm
459
460  Boot-BlockFile:
461     CLS
462     COLOR 3
463     PRINT "Boot-Block in File übertragen"
464     COLOR 1
465     PRINT
466     GOSUB Laufwerk
467     PRINT
468     Disk "r",0,Laufw%,Feld%()
469     If Fehler$<>"" THEN Fehlerausgabe
470     Out$=""
471     FOR I%=0 TO 511
472        Out$=Out$+CHR$(Feld%(I%))
473     NEXT
474     Disk "r",0,Laufw%,Feld%()
475     If Fehler$<>"" THEN Fehlerausgabe
476     FOR I%=0 TO 511
477        Out$=Out$+CHR$(Feld%(I%))
478     NEXT
479     INPUT "FileName:",FileName$
480     OPEN "o",#1,FileName$
481     PRINT#1,Out$;
482     CLOSE 1
483  GOTO Hauptprogramm
484
485  FileBlock:
486     CLS
487     COLOR 3
488     PRINT "File in Block übertragen"
489     COLOR 1
490     PRINT
```

```
491     INPUT "FileName:",FileName$
492     OPEN "i",#1,FileName$
493     If LOF(1)<>512 THEN
494        Fehler$="Filelänge <> 512"
495        CLOSE 1
496        GOTO Fehlerausgabe
497     END If
498     In$=INPUT$(512,1)
499     CLOSE 1
500     PRINT
501     GOSUB Laufwerk
502     PRINT
503     NeuE1:
504     PRINT
505     INPUT "Blocknummer (0-1759) :",I$
506     BlockNummer%=VAL(I$)
507     If BlockNummer%<0 OR BlockNummer%>1759 THEN NeuE1
508     FOR I%=0 TO 511
509        Feld%(I%)=ASC(MID$(In$,I%+1,1))
510     NEXT
511     Disk "w",BlockNummer%,Laufw%,Feld%()
512     If Fehler$<> "" THEN Fehlerausgabe
513  GOTO Hauptprogramm
514
515  BlockFile:
516     CLS
517     COLOR 3
518     PRINT "Block in File übertragen"
519     COLOR 1
520     PRINT
521     GOSUB Laufwerk
522     PRINT
523     NeuE2:
524     INPUT "Blocknummer (0-1759) :",I$
525     BlockNummer%=VAL(I$)
526     If BlockNummer%<0 OR BlockNummer%>1759 THEN NeuE2
527     PRINT
528     Disk "r",BlockNummer%,Laufw%,Feld%()
529     If Fehler$<>"" THEN Fehlerausgabe
530     Out$=""
531     FOR I%=0 TO 511
532        Out$=Out$+CHR$(Feld%(I%))
533     NEXT
534     INPUT "FileName:",FileName$
535     OPEN "o",#1,FileName$
536     PRINT#1,Out$;
537     CLOSE 1
538  GOTO Hauptprogramm
```

```
539
540  "***********************************************
541  "*                                             *
542  "*  Unterprogramm:  Disk                       *
543  "*                                             *
544  "*  Aufgabe:        Block lesen oder schreiben *
545  "*                                             *
546  "*  Parameter:      Modus$ "R" oder "W"        *
547  "*                  Block% 0-1759              *
548  "*                  Unit%  0-3                 *
549  "*                  Datenfeld%(511)            *
550  "*                                             *
551  "*  Ergebnis in:    Fehler$ ("" wenn OK)       *
552  "*                                             *
553  "*  Autor:          Mirko Langlotz             *
554  "*                                             *
555  "*  Datum:          18.09.1988                 *
556  "*                                             *
557  "***********************************************
558
559  SUB Disk(Modus$,Blk%,Unit%,DatenFeld%()) STATIC
560  SHARED Fehler$
561     If UCASE$(Modus$)="R" THEN
562        Comm%=2
563        Par&=512
564     ELSEIF UCASE$(Modus$)="W" THEN
565        Comm%=3
566        Par&=512
567     Else
568        Fehler$="Befehl nicht bekannt"
569        EXIT SUB
570     END If
571     If Unit%<0 OR Unit%>3 THEN
572        Fehler$="Falsche Laufwerksangabe"
573        EXIT SUB
574     END If
575     GOSUB CheckDrive
576     If Found%=0 THEN
577        Fehler$="Laufwerk nicht angeschlossen"
578        EXIT SUB
579     END If
580     If Blk%<0 OR Blk%>1759 THEN
581        Fehler$="Ungültige Blocknummer"
582        EXIT SUB
583     END If
584     GOSUB OpenTrackDisk
585     If Fehler$<>"" THEN EXIT SUB
586     If Comm%=3 THEN
```

```
587         FOR I%=0 TO 511
588            POKE Mem&+I%,DatenFeld%(I%)
589         NEXT
590      END If
591      BlockNr&=Blk%
592      GOSUB Action
593      MainComm%=Comm%
594      Fehler2$=Fehler$
595      Comm%=9
596      Par&=0
597      GOSUB Action
598      If Fehler2$<>"" THEN Fehler$=Fehler2$
599      If Fehler$="" AND MainComm%=2 THEN
600         FOR I%=0 TO 511
601            DatenFeld%(I%)=PEEK(Mem&+I%)
602         NEXT
603      END If
604      GOSUB CloseTrackDisk
605      EXIT SUB
606
607   "*****************************************
608   "*                                       *
609   "*  Unterprogramm: OpenTrackDisk         *
610   "*                                       *
611   "*  Aufgabe:       Zugriff vorbereiten   *
612   "*                                       *
613   "*  Anmerkung:     Wird automatisch von  *
614   "*                 "SUB Disk" aufgerufen.*
615   "*                                       *
616   "*****************************************
617
618   OpenTrackDisk:
619      Fehler$=""
620
621      OffsetPort& = 512
622      OffsetIO&   = OffsetPort&+50
623      OffsetName& = OffsetIO&+100
624
625      MSize&   = &H1000
626
627      Mem&=AllocMem&(MSize&,65536&+3)
628      If Mem&=0 THEN
629         Fehler$="Nicht genug Seicher"
630         Return
631      END If
632
633      DeviceName& = Mem&+OffsetName&
634      DeviceIO&   = Mem&+OffsetIO&
```

```
635     DevicePort& = Mem&+OffsetPort&
636
637     DevName$="trackdisk.device"+CHR$(0)
638     FOR I%=1 TO LEN(DevName$)
639        POKE DeviceName&+I%-1,ASC(MID$(DevName$,I%,1))
640     NEXT
641
642     Task&=FindTask&(0)
643     POKEL DevicePort&+16,Task&
644     POKEL DeviceIO&  +14,DevicePort&
645     AddPort(DevicePort&)
646
647     Status&=OpenDevice%(DeviceName&,Unit%,DeviceIO&,0)
648     GOSUB StatusError
649     If Status&<>0 THEN CALL FreeMem(Mem&,MSize&)
650  Return
651
652  "*****************************************
653  "*                                       *
654  "*  Unterprogramm: CloseTrackDisk        *
655  "*                                       *
656  "*  Aufgabe:       Zugriff beenden       *
657  "*                                       *
658  "*  Anmerkung:     Wird automatisch von  *
659  "*                 "SUB Disk" aufgerufen.*
660  "*                                       *
661  "*****************************************
662
663  CloseTrackDisk:
664     CloseDevice(DeviceIO&)
665     RemPort(DevicePort&)
666     CALL FreeMem(Mem&,MSize&)
667  Return
668
669  "*****************************************
670  "*                                       *
671  "*  Unterprogramm: StatusError           *
672  "*                                       *
673  "*  Aufgabe:       Meldung erstellen     *
674  "*                                       *
675  "*  Anmerkung:     Wird automatisch von  *
676  "*                 "SUB Disk" aufgerufen.*
677  "*                                       *
678  "*****************************************
679
680  StatusError:
681     If Status&=0 THEN
682        Fehler$=""
```

```
683     Else
684        RESTORE ErrorData
685        If Status&<20 OR Status&>35 THEN
686           Fehler$="Unbekannter Fehler"
687        Else
688           Fehler$="Disk-Fehler:"
689           FOR I%=1 TO Status&-19
690              READ R$
691           NEXT
692           Fehler$=Fehler$+STR$(Status&)+" "+R$
693        END If
694     END If
695  Return
696
697  ErrorData:
698     DATA NotSpecified
699     DATA NoSecHdr
700     DATA BadSecPreamble
701     DATA BadSecID
702     DATA BadHdrSum
703     DATA BadSecSum
704     DATA TooFewSecs
705     DATA BadSecHdr
706     DATA WriteProt
707     DATA DiskChanged
708     DATA SeekError
709     DATA NoMem
710     DATA BadUnitNumber
711     DATA BadDriveType
712     DATA DriveInUse
713     DATA PostReset
714
715  "*****************************************
716  "*                                       *
717  "*  Unterprogramm: Action                *
718  "*                                       *
719  "*  Aufgabe:       Block lesen/schreiben  *
720  "*                                       *
721  "*  Anmerkung:     Wird automatisch von   *
722  "*                 "SUB Disk" aufgerufen. *
723  "*                                       *
724  "*****************************************
725
726  Action:
727     POKEW DeviceIO&+28,Comm%
728     POKEL DeviceIO&+36,Par&
729     If Comm%=2 OR Comm%=3 THEN
730        POKEL DeviceIO&+40,Mem&
```

```
731         POKEL DeviceIO&+44,BlockNr&*512
732      END If
733      Status&=DoIO%(DeviceIO&)
734      If Status&=0 AND Comm%=3 THEN
735         POKEW DeviceIO&+28,4
736         Status&=DoIO%(DeviceIO&)
737      END If
738      GOSUB StatusError
739   Return
740
741   "*****************************************
742   "*                                       *
743   "*  Unterprogramm: CheckDrive            *
744   "*                                       *
745   "*  Aufgabe:       Laufwerk vorhanden?   *
746   "*                                       *
747   "*  Anmerkung:     Wird automatisch von  *
748   "*                 "SUB Disk" aufgerufen.*
749   "*                                       *
750   "*****************************************
751
752   CheckDrive:
753      ExecBase&=4
754      OffsetLibList&=378
755      LibList&=PEEKL(PEEKL(ExecBase&)+OffsetLibList&)
756      DosBase&=FindName&(LibList&,SADD("dos.library"
                            +CHR$(0)))
757      DevNameTab&=(PEEKL((PEEKL(PEEKL(DosBase&+34)
         +24)*4)+4)*4)
758      Found%=0
759      WHILE PEEKL(DevNameTab&)<>0
760         If PEEKL(DevNameTab&+4)=0 THEN
761            Help&=0" wichtig!
762            DriveName&=varprt(Help&)
763            POKE DriveName&+0,3
764            POKE DriveName&+1,ASC("D")
765            POKE DriveName&+2,ASC("F")
766            POKE DriveName&+3,ASC("0")+Unit%
767            If PEEKL((PEEKL(DevNameTab&+40)*4))=PEEKL
               (DriveName&) THEN Found%=1
768         END If
769         DevNameTab&=PEEKL(DevNameTab&)*4
770      WEND
771   Return
772
773   END SUB
774
```

Bedienungsanleitung

Der Disk-Monitor ist sehr einfach zu bedienen. Nachdem er abgetippt und auf Diskette gesichert ist, startet man ihn. Es erscheint eine erste Auswahl, in der man wählen kann, welche Funktion des Programms gewünscht wird. Es stehen dafür mehrere zur Verfügung:

Jeder Block der Diskette läßt sich mit dem Monitor editieren. Dazu wählt man die Nummer 1 an und gelangt in den Editor-Bereich. Hier wird im unteren Teil des Ausgabe-Windows der Inhalt des angewählten Blocks einerseits in hexadezimalen Zahlen und andererseits auch in ASCII-Code ausgegeben. Das eine gilt für Berechnungen, und mit Hilfe der Texte lassen sich leichter irgendwelche Bemerkungen ausfindig machen.

Sie können nun mit den Cursortasten eine beliebige Stelle auswählen und dort eine Eingabe machen. Entweder man schreibt einen Text, der an dieser Stelle eingefügt werden soll, dann kann dies ganz einfach über die Tastatur erfolgen, oder man möchte Daten, also Zahlen, eingeben.

Dafür wählt man den Eingabe-Modus an und kann einen hexadezimalen Zahlenwert an dieser Stelle eingeben. Danach ist man wieder im normalen Editier-Modus.

Des weiteren bietet der Disk-Monitor einige Funktionen für den Boot-Block, die es ermöglichen, diesen als Programm-File zu konvertieren oder ihn zu sichern.

Alles dies wurde aus der Überlegung der Viren heraus getan. Es ist beim Amiga möglich, ein Programm in den Boot-Block zu legen, das beim Starten des Amiga abgearbeitet wird. Auch Viren sind nur Programme, die im Boot-Block liegen und sich immer wieder neu dorthin schreiben.

Da diese Viren einen Boot-Block mit ihrem Hineinschreiben zerstören, gibt es zwei Menüpunkte, die einen Boot-Block sichern. Dazu überträgt man den Inhalt des Boot-Blocks in ein File und kann diesen später wieder zurück auf die Diskette schreiben. Es ist so möglich, von kommerzieller Software sozusagen Boot-Block-Backups zu machen, damit bei einer Zerstörung durch einen unerkannten Virus eine Rettung möglich ist.

Modulliste

```
SUB Disk(Modus$,Blk%,Unit%,DatenFeld%()) STATIC
```

Funktion

Diese übergreifende Subroutine ist für das Lesen oder Schreiben eines Blocks auf der Diskette zuständig. In ihr befinden sich alle Unterroutinen, die im weiteren Verlauf aufgelistet werden. Sie alle unterstützen die Subroutine bei ihrer Arbeit.

Parameter

Modus$ Read oder Write für Lesen oder Schreiben

Blk% Nummer des zu lesenden oder schreibenden Blocks

Unit% Nummer des anzusprechenden Laufwerks

DatenFeld%
Datenfeld, in dem die Daten des Blocks abgelegt werden sollen

OpenTrackDisk

Funktion

Dieses Unterprogramm öffnet das Trackdisk.Device, mit dem man auf einzelne Blöcke der Diskette zugreifen kann.

CloseTrackDisk

Funktion

Mit diesem Unterprogramm wird das zuvor geöffnete Trackdisk.Device wieder geschlossen. Das ist wichtig, denn jeder Zugriff auf ein Device muß zuvor angemeldet werden und kostet dabei Speicherplatz. Diesen sollte man wieder freigeben, wenn der Zugriff abgeschlossen ist, um auch den Speicher wieder freizugeben.

StatusError

Funktion

Diese Unterroutine ist für die Ermittlung eines aufgetretenen Fehlers zuständig. Sie gibt dazu aus den vorhandenen Fehlertexten den richtigen aus, um den Benutzer darüber zu informieren, welcher Fehler aufgetreten ist.

Action

Funktion

Mit diesem Unterprogramm werden die einzelnen Blöcke von der Diskette gelesen oder auf sie geschrieben. Hier passiert die eigentliche Arbeit des Disk-Monitors. Daher kommt auch der Name des Unterprogramms.

CheckDrive

Funktion

Diese letzte Unterroutine stellt fest, ob sich der angegebene Drive überhaupt im System befindet. Das heißt, es kann vorkommen, daß man ein Laufwerk angibt, das gar nicht existiert. Würde die Überprüfung nicht stattfinden, würde das Programm mit einem Fehler aussteigen oder abschmieren. Dies wird hierdurch vermieden.

5.3 Mit GFA-BASIC zum Erfolg

Das mittlerweile etablierte AmigaBASIC hat Ende 1988 Konkurrenz bekommen. GFA-BASIC heißt eine wirkliche Alternative, die neben dem BASIC-Standard von Microsoft, der nur etwa 30 Befehle umfaßt, ein wahrhaft enormes Spektrum von Befehlen enthält.

Dabei gibt es neben einer großen Anzahl von Grafik-Funktionen, auch sehr mächtige Befehle im Bereich Variablen-Organisation und Dateiverwaltung. Somit kann der Programmierer ebenfalls Unterstützung in den "ernsten" Bereichen erwarten. Als besondere Geste ist es zu werten, daß die Programmierer des GFA-BASIC die Animations-Befehle kompatibel zu denen des AmigaBASIC gehalten haben. Damit müssen gerade komplizierte Programmtexte, in denen Animationen durchgeführt werden, in bezug auf diese Befehle nicht umgeschrieben werden!

Der wirkliche Vorteil des GFA-BASIC ist aber nicht der Befehlsumfang, sondern die Geschwindigkeit, mit der die Programme abgearbeitet werden. Man kann sagen, daß diese der sehr verbreiteten Sprache C um fast nichts nachsteht. Somit wird es viel einfacher sein, Programme mit einem Interpreter zu schreiben, als sich langwierig durch einen Compiler zu quälen.

Um nun auch dieser "neuen" Programmiersprache gerecht zu werden, haben wir dieses Unterkapitel ab der zweiten Auflage ergänzt, in dem natürlich nicht die gesamte Programmiersprache abgedeckt werden kann. Wir wollen uns deshalb mit einem kleinen Einstieg begnügen, der es zum Ziel haben wird, Ihnen zum einen die Programmierumgebung mit all ihren Eigenschaften nahezubringen und zum anderen auf die Besonderheiten des neuen BASIC-Dialekts einzugehen.

5.3.1 GFA-BASIC stellt sich vor

Der Editor

Der GFA-BASIC-Editor arbeitet nach einem anderen Konzept als das AmigaBASIC. Man schreibt seine Programme nicht in einem Window auf der Workbench-Screen, das bei vielen offenen Windows ziemlich langsame Gra-

fikausgaben nach sich zieht. In GFA-BASIC wird ein separater Screen geöffnet, der abzüglich zwei Text-Zeilen vollständig zum Editieren zur Verfügung steht. Sehen Sie dazu die folgende Abbildung:

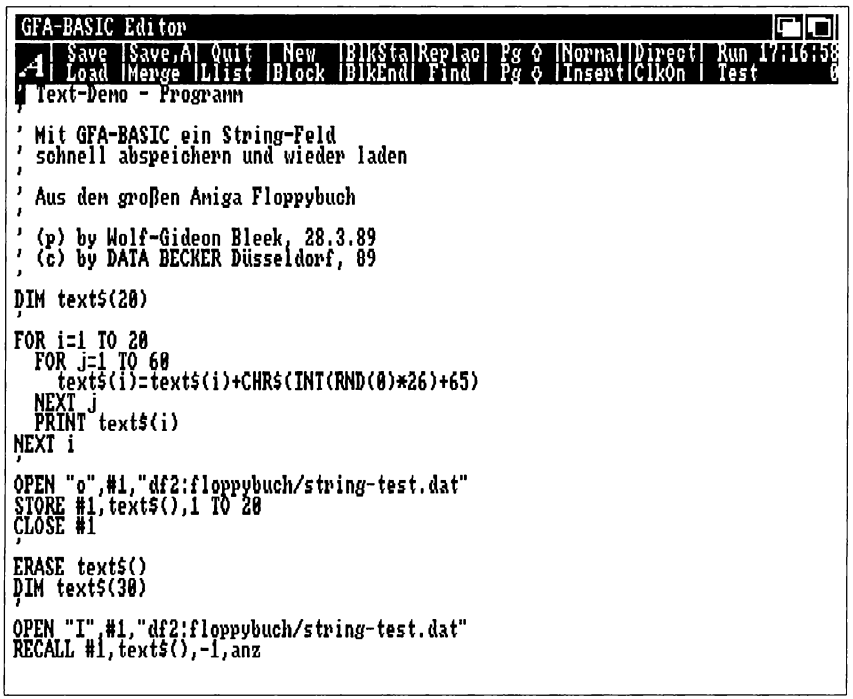


Abb. 5.3: Der GFA-BASIC Editor-Screen

In diesen oben erwähnten zwei Zeilen befinden sich alle Kommandos, die während der Arbeit gebraucht werden könnten. Außerdem steht hier eine Uhr in der Anzeige, so daß Sie niemals die Zeit verschlafen. Ein - wie wir finden - hilfreicher Einfall. Neben Befehlen zum Laden und Speichern in komprimierten und Text-Format werden hier Such- und Austausch-Hilfen gegeben, die es leichter machen, im Programm Änderungen vorzunehmen. Doch dazu jetzt im einzelnen:

Mit dem Cursor und den Editor-Befehlen BlkSta und BlkEnd läßt sich durch Festsetzen der Anfangs- und Endzeile eines Bereiches leicht ein Block markieren. Dieser kann nun leicht über das Untermenü "Block" modifiziert wer-

den. Dafür stehen neben dem bekannten Löschen und Verschieben noch weitere Funktionen zur Auswahl. Besonders interessant sind das Ausdrucken eines markierten Bereiches und das Schreiben des Bereiches auf Diskette. Somit können Unterroutinen leicht dokumentiert und als Bibliothek abgespeichert werden. In jedem neuen Programm können diese mit dem vom AmigaBASIC bekannten MERGE angehängt werden.

- Einstellungen:
  - Normal/Interlace
  - Insert/Overwrite
  - Clock On/Off
  - Zeilennummerneingabe
- Direkte Kommandos
- Syntax-Test, Run

Die File-Request-Box

Anders als alle uns bekannten Interpreter bietet das GFA-BASIC eine integrierte File-Auswahl, mit deren Hilfe viel Arbeit und Zeitaufwand verschwindet. Neben der Anzeige aller Files aus einem Verzeichnis erleichtert sie auch die Device-Auswahl, weil alle angeschlossenen Diskettenlaufwerke und ggf. Festplatten angezeigt werden.

Wie man aus der Abbildung erkennen kann, werden im oberen Bereich 10 Dateien, Programme oder Verzeichnisse angezeigt. Verzeichnisse erkennt man an dem vorangestellten Stern. Dies ist etwas gewöhnungsbedürftig. Unter dieser Auswahltabelle wird eine Liste aller Geräte ausgegeben, die für Diskettenoperationen ansprechbar sind. GFA-BASIC erkennt selbständig, welche Diskettenlaufwerke oder Festplatten angeschlossen sind.

Eine kleiner Wermutstropfen liegt allerdings in der Programmierung. Ist nämlich eine Diskettenoperation beendet und wird die Request-Box erneut aufgerufen, beginnt diese sofort wieder mit dem Laden des Verzeichnisses. Hier wäre es angenehmer gewesen, wenn das zuletzt geladene Verzeichnis gespeichert würde. Diese File-Request-Box wird nun aber nicht nur von LOAD,

SAVE und MERGE benutzt, sondern sie läßt sich auch vom Programm aus benutzen. Dafür gibt es einen einfachen Befehl, den wir Ihnen im Programmier-Teil vorstellen werden.

Der RUN-ONLY-INTERPRETER

Sehr angenehm ist der mitgelieferte GFA-BASIC RUN-ONLY-INTERPRETER (kurz GFAbasRO). Damit ist es möglich, die selbstgeschriebenen Programme auch bei Amiga-Usern laufen zu lassen, die nicht im Besitz des GFA-BASIC sind. Denn hier tritt anders als beim AmigaBASIC, das bei jedem Amiga mitgeliefert wird, das Problem auf, daß diese Programmiersprache käuflich zu erwerben ist. Der RO-Interpreter kann aber ohne Verletzung des Copyrights weitergegeben werden.

Hierbei fehlt lediglich der Editor. Deshalb ist dieses Programm nur in der Lage, im GFA-BASIC-Format abgespeicherte Programme zu lesen. Es nützt also nichts, mit einem ASCII-Editor zu arbeiten.

Die Menüs

Zusätzlich zur Befehlsleiste wird im Editor auch noch ein Menü angeboten, das alle wichtigen Funktionen der Leiste und noch einige Befehle zusätzlich enthält.

Hier findet man die Lade- und Speicherfunktionen genauso wieder wie den Befehl RUN zum Starten des Programms. Allerdings zeigt sich die Verbesserung im Detail. Dadurch, daß diese wiederholt wurden, können Sie auch über die Short-Cuts mit der [Amiga]-Taste erreicht werden.

In das Menü hinzugekommen sind einige Systemeinstellungen, die es z.B. ermöglichen, die Arbeit von GFA-BASIC noch weiter zu beschleunigen. Mit TASKPRI 1 wird das Multitasking dahingehend verändert, daß zu 90% GFA-BASIC abgearbeitet wird und andere im Hintergrund laufende Programme an letzter Stelle stehen. Dies läßt sich aber jederzeit mit TASKPRI 0 wieder abschalten.

Nützlich ist auch die Möglichkeit, mit NewCLI ein neues CLI-Fenster zu öffnen, um so schnell Diskettenoperationen ausführen zu können. Leider wurde in der uns vorliegenden Version die neue Shell noch nicht unterstützt.

5.3.2 Strukturiert programmieren leichtgemacht

Neben den Befehlen, die die eigentlichen Aktionen hervorrufen wie z.B. Zeichenbefehle oder andere zum Ausgeben von Tönen, gibt es noch die Gruppe der Struktur-Befehle, die den Programmcode durch organisatorische Gestaltung besser lesbar machen.

Das GFA-BASIC bietet hier neben den auch in AmigaBASIC vertretenen Schleifentypen noch einiges mehr. Ein ganz besonders hilfreiches Feature ist das automatische Einrücken, wodurch sich die Verschachtelungstiefen ganz automatisch darstellen und man Fehler ganz leicht entdeckt. Sehen Sie dazu als Beispiel eine IF...THEN-Folge, die so, wie sie abgedruckt wurde, vom Editor eingerückt worden ist.

```
" Programm zur Demonstration
" der IF .. THEN .. ELSE
" Konstruktion
"
PRINT "Bitte Geben Sie eine Zahl ein!"
INPUT zahl
"
IF zahl=0 THEN
  PRINT "Sie haben Null eingegeben."
ELSE
  IF zahl>0
    PRINT "Die ist größer als Null!"
    IF zahl>100
     PRINT "Die Zahl ist sogar wesentlich größer als Null!"
    ENDIF
  ELSE
    PRINT "Die Zahl ist ja kleiner als Null!"
    IF zahl<-100
      PRINT "Die Zahl ist viel zu klein!"
    ENDIF
  ENDIF
ENDIF
"
PRINT "Tschüß"
```

Neben der Unterscheidung von verschiedenen Fällen kommt es innerhalb der Programmentwicklung wesentlich häufiger vor, daß man bestimmte Aktionen sehr oft wiederholen muß. So z.B. die Berechnung eines Wertes, bis eine gewünschte Genauigkeit erreicht worden ist. Dazu bieten sich Schleifen an, die unter einer festgelegten Bedingung wiederholt oder aber abgebrochen werden.

Hier sehen Sie eine Tabelle der Schleifen und ihrer Eigenschaften:

Schleife	Eigenschaft
FOR TO NEXT	Feste Anzahl von Durchläufen
DO LOOP	Unendlich (oder Abbruch mit Bedingung)
REPEAT UNTIL	Abbruchbedingung mit Endüberprüfung
WHILE WEND	Abbruchbedingung mit Anfangsüberprüfung

Jetzt sehen Sie ein und dasselbe Programm in den unterschiedlichen Schleifentypen realisiert. Natürlich wirken einigen Methoden umständlich, denn nicht jede Variante kann immer ohne Probleme angewendet werden.

```
FOR TO NEXT:
" Ausgabe der ersten Quadratzahlen
"
FOR i=0 TO 10
  PRINT i;"2=";i*i
NEXT i
DO LOOP:
" Ausgabe der ersten Quadratzahlen
"
i=0
DO
  PRINT i;"2=";i*i
  EXIT IF i=10
  INC i
LOOP
REPEAT UNTIL:
" Ausgabe der ersten Quadratzahlen
"
i=0
REPEAT
  PRINT i;"2=";i*i
  INC i
UNTIL i>10
WHILE WEND:
```

```
" Ausgabe der ersten Quadratzahlen
"
i=0
WHILE i<=10
  PRINT i;"2=";i*i
  INC i
WEND
```

Wie sich leicht an diesem Beispiel erkennen läßt, bietet für dieses "Problem" die FOR TO NEXT-Schleife die beste Lösung. Allerdings versagt sie bei Programmsegmenten, die eine - zu Anfang - unbestimmte Anzahl von Druchläufen bearbeitet werden müssen.

5.3.3 Mit superschneller Grafik zum Erfolg

Eine weitere Stärke des GFA-BASIC ist die Grafik-Programmierung. Hier wurden bei der Amiga-Version die Befehle vom Atari ST übernommen. Somit wurde zwar keine Kompatibilität erreicht (der Atari ST besitzt ganz andere Grafik-Modi), aber die meisten Befehle ähneln sich sehr. Dies ist besonders für diejenigen interessant, die auf den Amiga umgestiegen sind.

Am folgenden Beispiel wollen wir zeigen, wie schnell Grafik sein kann. Das Programm zeichnet eine Linie, deren X- und Y-Koordinaten-Paare zufällig verändert werden. Man kennt dieses Demo unter dem Namen Quix. Schon an diesem Beispiel läßt sich erkennen, wie schnell ein BASIC-Programm sein kann. Wenn Sie mit ein wenig Änderung das Programm in Amiga BASIC laufen lassen, zeigt sich der Geschwindigkeitsunterschied ganz deutlich. Dies liegt vor allem an den äußerst mächtigen Befehlen wie z.B. INSERT, der automatisch ein neues Element in ein Array einfügt.

```
" Zufällige Bewegung einer Linie
" mit zeitverzögerter Löschung
"
" Das große Amiga 500 Buch
"
" (p) by Wolf-Gideon Bleek August 1989
" (c) by DATA BECKER GmbH August 1989
"
dimens=20           ! Anzahl der gespeicherten Koordinaten
aktion=2            ! Stärke der Bewegung in Pixeln
range=aktion*2+1    ! Bereich der Zufallszahlen
```

```
"
DIM x1(dimens),x2(dimens),y1(dimens),y2(dimens)
"
x1(0)=RANDOM(640)    ! Linienposition festlegen
x2(0)=RANDOM(640)
y1(0)=RANDOM(256)
y2(0)=RANDOM(256)
"
WHILE TRUE
  COLOR 1
  LINE x1(0),y1(0),x2(0),y2(0)
  COLOR 0
  LINE x1(dimens),y1(dimens),x2(dimens),y2(dimens)
  INSERT x1(0)=x1(0)+(RANDOM(range)-aktion)
  INSERT x2(0)=x2(0)+(RANDOM(range)-aktion)
  INSERT y1(0)=y1(0)+(RANDOM(range)-aktion)
  INSERT y2(0)=y2(0)+(RANDOM(range)-aktion)
  x1(0)=x1(0)*SGN(x1(0))
  IF x1(0)>640 THEN
    x1(0)=640
  ENDIF
  x2(0)=x2(0)*SGN(x2(0))
  IF x2(0)>640 THEN
    x2(0)=640
  ENDIF
  y1(0)=y1(0)*SGN(y1(0))
  IF y1(0)>256 THEN
    y1(0)=256
  ENDIF
  y2(0)=y2(0)*SGN(y2(0))
  IF y2(0)>256 THEN
    y2(0)=256
  ENDIF
WEND
```

Listing 5.3.2.1: Quix.GFA

Auch zu einer anderen Demonstration eignet sich dieses Programm sehr gut. Schalten Sie nämlich vor dem Start im Menü die TASKPRI 1 ein, werden andere Programme nicht mehr bearbeitet. Der Quix läuft fröhlich schnell vor sich hin, aber es ist nicht mehr möglich, auf der Workbench eine Schublade zu öffnen. Erst wenn Sie das Programm mit Alt-Shift-Ctrl unterbrechen wird plötzlich wie von Geisterhand das Unterverzeichnis aufgemacht.

Versuchen Sie als nächstes einmal das folgende Beispiel, in dem wiederum eines der Grafik-Demos, die von Commodore bei der Version 1.2 der Workbench auf der Extras-Diskette mitgeliefert wurden, nachprogrammiert worden ist. Hierbei haben wir es mit Kästen zu tun, einer grafischen Spezialität des Amiga. Dadurch, daß für grafische Grundoperationen wie das Füllen von rechteckigen Flächen ein gesonderter Chip, Denise, eingebaut wurde, gibt es auf diesem Gebiet nichts Vergleichbares. Allerdings muß die Aussteuerung gut geregelt sein, damit keine Zeit verloren geht.

```
" Zufällige Rechtecke im
" Ausgabe-Window
"
" (p) by Wolf-Gideon Bleek August 1989
" (c) by DATA BECKER GmbH August 1989
"
WHILE TRUE
  x1=RANDOM(640)
  x2=RANDOM(640)
  y1=RANDOM(256)
  y2=RANDOM(256)
  COLOR RANDOM(4)
  PBOX x1,y1,x2,y2
WEND
```

Listing 5.3.2.2: Boxes

Das Programm zeichnet in das Ausgabe-Window in einer derartig hohen Geschwindigkeit Rechtecke, daß man eigentlich nicht glauben kann, daß es sich um ein BASIC-Programm handelt. Allerdings haben wir diese Funktion damit noch nicht ausgereizt. Durch die Definition eines Füllmusters wird es ja erst interessant:

```
" Zufällige Rechtecke im
" Ausgabe-Window mit verschiedenen
" Füllmustern
"
" Das große Amiga 500 Buch
" Das große Amiga 500 Buch
" (p) by Wolf-Gideon Bleek August 1989
" (c) by DATA BECKER GmbH August 1989
"
WHILE TRUE
  DEFFILL RANDOM(4),2,RANDOM(24)
  x1=RANDOM(640)
```

```
    x2=RANDOM(640)
    y1=RANDOM(256)
    y2=RANDOM(256)
    COLOR RANDOM(4)
    PBOX x1,y1,x2,y2
WEND
```

Listing 5.3.2.3: Filled Boxes

Obwohl das Listing nur um eine einzige Zeile ergänzt wurde (der Befehl DEFFILL legt das Füllmuster fest), ergibt sich eine vollkommen neue Perspektive. Denn bei gleicher Geschwindigkeit ist das BASIC-Programm zusätzlich in der Lage, zweifarbige Füllmuster zu verwenden.

Innerhalb des Programms benutzen wir mit

```
DEFFILL RANDOM(4),2,RANDOM(24)
```

den Mustersatz 2. Dieser wird in einer von vier zufällig gewählten Farben gezeichnet. Auch das Muster selbst ist zufällig. Es gibt hier insgesamt 24 verschiedene Strukturen. Der Mustersatz 3 enthält dagegen nur 12.

Interessant sind neben den Zeichenbefehlen auch die Animations-Kommandos. Beide, Sprites und Bobs, werden wie im AmigaBASIC auch mit den gleichen Befehlen gehandhabt. Es gibt allerdings auch schon fertige Sprite-Befehle, die noch aus der Zeit des Atari ST stammen.

5.3.4 System-Routinen standardmäßig nutzen

Ähnlich wie das AmigaBASIC besitzt auch das GFA-BASIC eine Schnittstelle zum Betriebssystem, mit der man an alle Libraries und somit auch an alle Funktionen des Systems herankommt. Sollte es irgendeinen Befehl nicht geben oder dieser nicht die benötigten Funktionen aufweisen, ist es ohne weiteres möglich, auf die Bibliotheken zuzugreifen.

Allerdings wurden anders als beim AmigaBASIC die wichtigsten Libraries schon von Grund auf integriert. So kann man die Exec-, die DOS-, die Graphics- und die Intuition-Library ganz selbstverständlich nutzen. Man muß sie noch nicht einmal öffnen.

Das liegt vor allem daran, daß die Programmierer sich gesagt haben: "Wenn wir sowieso die Libraries für das GFA-BASIC öffnen, dann kann auch gleich jeder auf diese Funktionen zugreifen".

1. Speicherverwaltung

Will man die System-Routinen nutzen, ist ein äußerst wichtiger Aspekt die Speicherverwaltung. Denn im Gegensatz zu den BASIC-Befehlen, die sich Daten aus Variablen holen und dort hineinschreiben, bieten die System-Routinen diesen Komfort nicht. Man muß selbst dafür sorgen, genügend Speicher zu haben, damit die Funktionen ordnungsgemäß laufen. Deshalb gibt es eigens zwei Funktionen, die für die Speicherverwaltung vorgesehen sind.

MALLOC (kurz für Memmory Allocation) besorgt einen Speicherbereich in der gewünschten Größe und liefert als Ergebnis die Speicheradresse.

MFREE (kurz für Memory free) löscht den vorher besorgten Speicherbereich wieder und gibt ihn zur allgemeinen Verwendung frei.

Das folgende Programm besorgt nach der Abfrage des Speichertyps und der Speichergröße einen solchen Speicherbereich. Gleich darauf wird er wieder in den Speicherpool des Systems freigegeben:

```
" Programm zur Demonstration
" der Speicherverwaltung
"
" (p) by Wolf-Gideon Bleek August 1989
" (c) by DATA BECKER GmbH August 1989
"
INPUT "Wie groß soll der Speicher sein (in Bytes)";size
INPUT "Welcher Speichertyp (2=CHIP, 4=FAST)";typ
"
adr=MALLOC(size,typ)
IF adr THEN
  PRINT "Geschaft! Der Speicher ist unser!"
ELSE
  PRINT "Leider gibt es den geforderten Speicher nicht"
ENDIF
back=MFREE(adr,size)
```

Listing 5.3.3.1: Malloc

2. Mitten ins System

Ist die Speicherverwaltung erst einmal sichergestellt, können alle anderen Arbeiten innerhalb der Systemprogrammierung in Angriff genommen werden. Der Aufruf und die Verarbeitung gestaltet sich genauso wie auch im Amiga-BASIC. Einziger Unterschied ist der fehlende Library-Befehl und die dazugehörenden Deklarationen. Dafür müssen wir allerdings einen anderen Standard erfüllen. Anhand des nächsten Beispiels wollen wir zeigen, wie man z.B. eine Liste aller im System geöffneten Screens und Windows ausgeben kann. Dafür ist ein gar nicht so aufwendiges Programm notwendig.

```
" Programm zur Ermittlung der im System
" integrierten Screens und Windows
"
" Das große 500`er Buch
"
" (p) by Wolf-Gideon Bleek im August 1989
" (c) by DATA BECKER GmbH 1989
"
DEFINT "a-z"
scrptr%=LPEEK(_IntBase+60)     ! IntuitionBase->FirstScreen
WHILE scrptr%<>0
  PRINT "SCREEN-Titel: ";
  @printtext(LPEEK(scrptr%+22))     ! Screen->Title
  PRINT "Default Titel: ";
  @printtext(LPEEK(scrptr%+26))     ! Screen->DefaultTitle
  wndptr%=LPEEK(scrptr%+4)
  WHILE wndptr%<>0
    PRINT "   WINDOW-Titel: ";
    @printtext(LPEEK(wndptr%+32))
    PRINT
    wndptr%=LPEEK(wndptr%)
  WEND
  scrptr%=LPEEK(scrptr%)
WEND
"
PROCEDURE printtext(zeiger%)
  WHILE PEEK(zeiger%)<>0
    PRINT CHR$(PEEK(zeiger%));
    INC zeiger%
  WEND
  PRINT
Return
```

Listing 5.3.3.2: Screen-Liste

Programmbeschreibung

Zuerst wird über die IntuitionBase der Zeiger auf den ersten Screen geholt. Dies ist nicht, wie man zuerst denken mag, der zuerst geöffnete Screen, sondern derjenige, der im Vordergrund steht. Der Name wird ausgegeben, und in einer Unter-Schleife untersucht das Programm noch, ob sich dort Windows befinden. Dies ist natürlich der Fall, denn auf dem Workbench-Screen öffnet das GFA-BASIC ja das Ausgabefenster. Also wird auch eine Liste aller Fenster ausgegeben.

Danach erfolgt wie auch bei den Fenstern die Weitersetzung des Screen-Zeigers auf den darauffolgenden. Sollte dieser Zeiger gleich Null sein, ist das Ende der Liste erreicht, und die Ausgabe kann erfolgreich abgebrochen werden.

Wenn Sie mit diesem Programm etwas experimentieren möchten, können wir Ihnen nur empfehlen, mit GFA-BASIC weitere Screens zu öffnen. Damit wird das Ausgabe-Fenster nicht mehr auf dem Workbench-Screen geöffnet, und es ergibt sich eine ganz andere Liste.

Um die Anzahl der Fenster zu steigern, reicht es aus, auf der Workbench einige Verzeichnisse zu öffnen. Besonders gut eignet sich dieses Programm aber auch dazu, herauszufinden, mit wie vielen Fenstern andere Programme auf ihren eigenen Screens arbeiten. Wieviel Fenster benutzt z.B. DeluxePaint?

Zum Schluß noch ein Tip!

Wenn Sie sich wirklich für die internen Parameter und System-Daten interessieren und speziell von Intuition mehr wissen möchten, sei Ihnen zum einen die Lektüre des großen C-Buchs von DATA BECKER empfohlen, in dem ausführlich die Programmierung von Intuition beschrieben wird. Sollten Sie sich damit schon ausreichend auskennen, können wir zur Unterstützung noch auf das Amiga Intern Band 2 hinweisen, in dem alle Funktionen und Strukturen zum Nachschlagen dokumentiert wurden.

Mit diesem Wissen sollte es dann keine Schwierigkeit mehr darstellen, z.B. die Liste der Menüs zu jedem Window auszugeben oder aber die Gadgets genauer zu erforschen. Im Prinzip sind Ihnen dabei keine Grenzen gesetzt. Es sei

nur die Empfehlung ausgesprochen, daß mit wachsender Programmgröße ein Auswahlmenü hinzugefügt werden sollte, daß es dem Anwender ermöglicht, ganz bestimmte Daten abzurufen.

Kommen wir jetzt zu unserem nächsten Beispielprogramm, das eine neue System-Einheit benutzt. Die Rede ist von "Exec", dem System-Verwaltungsprogramm. Diese Library ist für alle Programme und deren Kommunikation verantwortlich. Hier sind über alles Informationen vorzufinden, was im System gerade als Programm abläuft.

Das spezielle Listing, was wir hier abdrucken, gibt nur einen kleinen aber sehr interessanten Einblick in die Programmverwaltung des Multi-Tasking-Betriebssystems. Obwohl der Computer nur ein Programm zur Zeit abarbeiten kann, werden beim Amiga ja beliebig viele Programme gleichzeitig verarbeitet (was sich in der abnehmenden Geschwindigkeit bemerkbar macht). Das Prinzip ist einfach: jedes Programm erhält nur einen Teil der Prozessorzeit. Der Rest wird auf alle anderen Programme verteilt. Diese Verteilungsarbeit übernimmt das "Exec".

Die dafür verwendete Liste aller Prozesse lassen wir uns nun mit dem folgenden Listing ausgeben. Für weitere System-Informationen möchten wir Sie noch einmal auf das Amiga Intern Band 2 verweisen, in dem auch die "Exec"-Strukturen dokumentiert wurden.

```
" Programm zur Ermittlung der im System .
" laufenden Tasks
"
" Das große 500`er Buch
"
" (p) by Wolf-Gideon Bleek im August 1989
" (c) by DATA BECKER GmbH 1989
"
DEFINT "a-z"
DIM tasks%(50)
nr%=1
"
sysbase%=LPEEK(&H4)
tasks%(nr%)=LPEEK(sysbase%+276)       ! SysBase->ThisTask
INC nr%
"
VOID Disable()
"
task%=LPEEK(sysbase%+420)    ! SysBase->TaskWait
```

```
WHILE LPEEK(task%)<>0
  tasks%(nr%)=task%          ! Node=Task
  INC nr%
  task%=LPEEK(task%)         ! Task = Task->tc_Node.ln_Succ
WEND
"
task%=LPEEK(sysbase%+406)    ! SysBase->TaskReady
WHILE LPEEK(task%)<>0
  tasks%(nr%)=task%          ! Node=Task
  INC nr%
  task%=LPEEK(task%)         ! Task = Task->tc_Node.ln_Succ
WEND
"
VOID Enable()
"
PRINT "Es gibt ";nr%;" Tasks im System"
PRINT
PRINT TAB(1);"Adresse";TAB(10);"Name";TAB(30);"Status"
FOR i%=1 TO nr%-1
  PRINT tasks%(i%);TAB(10);
  text%=LPEEK(tasks%(i%)+10)
  WHILE PEEK(text%)<>0
    PRINT CHR$(PEEK(text%));
    INC text%
  WEND
  PRINT TAB(30);
  status%=PEEK(tasks%(i%)+15)
  IF status%=1
    PRINT "ADDED"
  ELSE IF status%=2
    PRINT "RUN"
  ELSE IF status%=3
    PRINT "READY"
  ELSE IF status%=4
    PRINT "WAIT"
  ELSE IF status%=5
    PRINT "except"
  ELSE IF status%=6
    PRINT "removed"
  ELSE
    PRINT status%
  ENDIF
NEXT i%
```

Listing 5.3.3.3: Task-Liste

Eine Beispielausgabe:

```
Es gibt 15 Tasks im System

 Adresse  Name                 Status
12777512  GFABASIC:GFABASIC    RUN
12681288  alf.device           WAIT
12630672  DH0                  WAIT
12842032  RAD                  WAIT
12839368  RAM                  WAIT
12610808  File System          WAIT
12606976  File System          WAIT
12585624  File System          WAIT
12588102  trackdisk.device     WAIT
12601318  trackdisk.device     WAIT
12612142  trackdisk.device     WAIT
12592818  input.device         WAIT
12754008  Workbench            WAIT
12773312  PopCLI III           WAIT
```

Wie man sieht, ist im System einiges mehr los, als man es zuerst erwartet. Die Ausgabe wurde hier allerdings etwas geordnet. Die Reihenfolge hängt immer gerade davon ab, wann man das Programm startet.

Als aktiver und laufender Task findet man natürlich immer das Programm vor, von dem man diese Liste anfertigen läßt. Danach folgen allerdings einige Programme, die weniger bekannt sein dürften. Das Programm "alf.device" gehört zu dem darauffolgenden "DH0". Diese beiden Tasks regeln die Festplattenverwaltung in meinem Amiga. RAM und RAD sind Verwaltungsprogramme für die beiden RAM-Disks. Für jedes angeschlossene Diskettenlaufwerk gibt es einen weiteren "File System"- und "trackdisk.device"-Task, wodurch sich leicht erkennen läßt, das insgesamt drei Diskettenlaufwerke angeschlossen sind. Das "input.device" ist für die Eingabe zuständig und steht in Zusammenhang mit der geöffneten Workbench, die auch als Task, also als Programm, vorliegt. Das einzig ungewöhnliche ist das Programm "PopCLI III", das vielleicht nicht allen bekannt sein wird.

Es gehört zum Lieferumfang des Lattice-C-Compiler Paketes. Dieses Tool schaltet nach einer wählbaren Zeit den Bildschirm ab und verhindert so, daß die Grafik einbrennt. Außerdem läßt sich mit seiner Hilfe jederzeit ein Programm wie z.B. die "Shell" aufrufen, mit dem man sich mehrere Workbench-Operationen spart.

Programmbeschreibung

Das Programm verwendet die Variablen tasks%() als Feld, in dem die Zeiger auf alle aktiven Tasks aufbewahrt werden, und nr% als Zähler, der die Anzahl der Tasks speichert.

Der weitere Programm-Code trägt nun nacheinander die verschiedenen Task-Listen in das Feld ein. Zuerst der aktive Task, der natürlich das GFA-BASIC ist. Als nächstes folgt die Liste aller wartenden Tasks und die aller Tasks, die bereit sind, als nächstes abgearbeitet zu werden. Um diese Ausleseschleifen wurden die Befehle Disable() und Enable() gesetzt, damit in dieser Zeit das Task-Switching nicht weiterläuft und die Liste während des Auslesevorgangs verändert wird.

Ist die Task-Liste komplett, wird dieser in einer weiteren Schleife ausgegeben. Und zwar erfolgt dies in einer Tabelle, die aus der Adresse der Task-Struktur, dem Namen des Tasks und seinem Zustand besteht. Für Interessierte ist es natürlich ohne weiteres noch möglich, andere Daten wie z.B. den Wert Task-> tc_Flags (14) oder Task->tc_SPUpper-Task->tc_SPLower (62/58) abzufragen, womit Sie weitere Informationen über den Zustand des Tasks erfahren bzw. die Größe des Task-Stacks ermitteln können.

3. Den Systemzugriff erweitern

Zwar unterstützt das GFA-BASIC die Handhabung der wichtigsten System-Libraries ohne Probleme, doch kommt man damit alleine nicht immer zurecht. Deshalb ist es auch ohne Probleme möglich, Zugriff auf weitere Libraries zu erhalten. Dafür wird einfach die Funktion OpenLibrary() der Exec-Library aufgerufen. Das folgende Beispielprogramm öffnet und schließt gleich darauf wieder die Icon.Library:

```
" Programmbeispiel: Wie öffnet man eine
" Library
"
" Das große Amiga 500 Buch
"
" (p) by Wolf-Gideon Bleek im September 1989
" (c) by DATA BECKER GmbH 1989
"
DEFINT "a-z"
```

```
version%=33
libname$="icon.library"
"
lib%=OpenLibrary(VARPTR(libname$),version%)
"
PRINT "Die Basisadresse der ";libname$;" ist ";lib%;"."
"
IF lib%<>0
  VOID CloseLibrary(lib%)
ENDIF
```

Die Funktion OpenLibrary() benötigt für die korrekte Arbeit zwei Parameter. Als erstes den Namen der Library, der als Zeiger (Speicheradresse) auf den Text übergeben wird und als zweites die Versionsnummer der Library. Hier wurde 33 gewählt. Dies entspricht der Workbench Version 1.2. Zurückgegeben wird die Basisadresse der Library, die wir hier in lib% festhalten. Diesen Variablennamen müssen Sie entsprechend der Library wählen! Darauf ist bei der Programmierung zu achten. Weiterhin ist ganz besonders darauf zu achten, daß die Library nur wieder geschlossen wird, wenn vorher das Öffnen ohne Probleme erfolgt ist. Sonst stürzt der Amiga hoffnungslos ab.

Als nächstes benötigen wir die Library-Funktionen selbst. Diese sind zwar nach dem Öffnen im System vorhanden, können aber ohne Hilfestellung nicht vom GFA-BASIC angesprochen werden. Auf der Diskette wird deshalb ein sog. FD-Converter mitgeliefert, der ohne Probleme diese Arbeit für uns erledigt.

Sehen Sie hier die konvertierten Funktionen der Icon.Library:

```
PROCEDURE getwbobject(name%)
  m68%(8)=name%
  m68%(14)=iconbase%
  RCALL iconbase%-30,m68%()
Return
PROCEDURE putwbobject(name%,object%)
  m68%(8)=name%
  m68%(9)=object%
  m68%(14)=iconbase%
  RCALL iconbase%-36,m68%()
Return
PROCEDURE geticon(name%,icon%,freelist%)
  m68%(8)=name%
  m68%(9)=icon%
  m68%(10)=freelist%
```

```
  m68%(14)=iconbase%
  RCALL iconbase%-42,m68%()
Return
PROCEDURE puticon(name%,icon%)
  m68%(8)=name%
  m68%(9)=icon%
  m68%(14)=iconbase%
  RCALL iconbase%-48,m68%()
Return
PROCEDURE freefreelist(freelist%)
  m68%(8)=freelist%
  m68%(14)=iconbase%
  RCALL iconbase%-54,m68%()
Return
PROCEDURE freewbobject(wbobject%)
  m68%(8)=wbobject%
  m68%(14)=iconbase%
  RCALL iconbase%-60,m68%()
Return
PROCEDURE allocwbobject
  m68%(14)=iconbase%
  RCALL iconbase%-66,m68%()
Return
PROCEDURE addfreelist(freelist%,mem%,size%)
  m68%(8)=freelist%
  m68%(9)=mem%
  m68%(10)=size%
  m68%(14)=iconbase%
  RCALL iconbase%-72,m68%()
Return
PROCEDURE getdiskobject(name%)
  m68%(8)=name%
  m68%(14)=iconbase%
  RCALL iconbase%-78,m68%()
Return
PROCEDURE putdiskobject(name%,diskobj%)
  m68%(8)=name%
  m68%(9)=diskobj%
  m68%(14)=iconbase%
  RCALL iconbase%-84,m68%()
Return
PROCEDURE freediskobject(diskobj%)
  m68%(8)=diskobj%
  m68%(14)=iconbase%
  RCALL iconbase%-90,m68%()
Return
PROCEDURE findtooltype(tooltypearray%,typename%)
  m68%(8)=tooltypearray%
```

```
  m68%(9)=typename%
  m68%(14)=iconbase%
  RCALL iconbase%-96,m68%()
Return
PROCEDURE matchtoolvalue(typestring%,value%)
  m68%(8)=typestring%
  m68%(9)=value%
  m68%(14)=iconbase%
  RCALL iconbase%-102,m68%()
Return
PROCEDURE bumprevision(newname%,oldname%)
  m68%(8)=newname%
  m68%(9)=oldname%
  m68%(14)=iconbase%
  RCALL iconbase%-108,m68%()
Return
```

Hierbei handelt es sich um das icon_lib.fd-File der Version 1.3 des Betriebssystems. Wie sich unschwer erkennen läßt, werden die Prozeduren dazu benutzt, die übergebenen Parameter in ein Feld namens m68%() einzutragen, die Funktion über die Basis iconbase% aufzurufen und danach in den normalen Programmablauf wieder zurückzukehren.

Aus diesem Grund müssen Sie innerhalb Ihres Programms folgende Vorkehrungen treffen:

1. Definieren Sie das Feld m68%() mit 16 Elementen

2. Legen Sie das Ergebnis der Funktion OpenLibrary() in der Variablen iconbase% ab bzw. analog dazu in den jeweiligen anderen Basis-Namen der Library.

3. Ergänzen Sie die Funktionsdefinitionen zur angewendeten Library mit MERGE am Programm-Ende.

4. Ändern Sie alle Prozeduren zu Funktionen, bei denen Sie das Ergebnis weiterverarbeiten möchten. Da dieser Prozeß etwas komplizierter ist, wollen wir Ihnen dies an einem Beispiel einmal vorführen:

Die hier abgedruckte Funktion GetDiskObject liefert als Ergebnis den Zeiger auf das Disk-Objekt zurück. Diesen findet die Funktion im Feld unter dem Index 0 wieder.

```
PROCEDURE getdiskobject(name%)
  m68%(8)=name%
  m68%(14)=iconbase%
  RCALL iconbase%-78,m68%()
Return
```

Wir ergänzen deshalb die Return-Zeile wie folgt:

```
Return m68%(0)
```

Des weiteren wird die Prozedur in eine Funktion umgewandelt und hat dann das folgende Aussehen:

```
FUNCTION getdiskobject(name%)
  m68%(8)=name%
  m68%(14)=iconbase%
  RCALL iconbase%-78,m68%()
  Return m68%(0)
ENDFUNC
```

6. Neue Welten - andere Programmiersprachen

Neben dem AmigaBASIC, das zum Amiga 500 kostenlos mitgeliefert wird, gibt es natürlich noch andere Programmiersprachen, die das gleiche Anrecht auf Erwähnung haben. Aber nicht nur dieses Recht macht andere Programmiersprachen interessant. Bevor wir uns den speziell hier ausgewählten Sprachen widmen, sollten wir zuerst einmal feststellen, was eigentlich Programmiersprachen ganz allgemein voneinander unterscheiden kann.

Es gibt dafür verschiedene Standpunkte, von denen man ausgehen kann. Der eine Betrachtungspunkt ist die "Nähe" zum Prozessor selbst. Ein anderer die Art die Gliederung, der Programme einer speziellen Sprache unterliegen. Ein dritter Punkt ist die Art der Ausführung von Befehlen. Gehen wir diese drei wichtigsten Punkte durch.

Jede Computersprache ist im eigentlichen Sinne nicht vom Computer verstehbar. Es wird in fast allen Fällen ein Programm gebraucht, das die für Menschen mehr oder minder verständlichen Worte in für den Computer verständliche Zahlen-Zustände übersetzt. Je nach "Schwere" der Übersetzung (Aufwand, Zeit und Kompliziertheit) wird die Nähe zum Prozessor gewertet.

So ist es sicherlich einfacher, einen Maschinenbefehl über eine Tabelle in die entsprechende Zahl zu übersetzen, als ein BASIC-Kommando in eine Folge von Maschinenbefehlen und dann in die für den Chip verständlichen Zahlen.

Hier sieht man auch gleich, daß ein Maschinenbefehl zu einer Zahl führt, die der Computer versteht. Ein BASIC-Befehl führt im Endeffekt zu mehreren Zahlen. Es gibt also zusätzlich einen Grad der Komplexität, der zwei Sprachen unterscheidet. Je komplexer eine Sprache ist, desto einfacher ist sie im allgemeinen für den Anwender verständlich. Andererseits ist es dann auch viel schwerer für das Übersetzungsprogramm, die richtige Übersetzung zu finden.

Bei der Übersetzung selbst gibt es auch unterschiedliche Methoden. Von BASIC wird die Simultanmethode bevorzugt. Das heißt, daß ein Programm in dem Moment, in dem es gebraucht wird, auch übersetzt werden muß. Das hat

einerseits zur Folge, daß man jederzeit schnell Veränderungen vornehmen kann. Das macht ein Programm aber andererseits langsamer, denn in Schleifen müssen die inneren Befehle bei jedem Durchlauf neu übersetzt werden.

An diesem kritischen Punkt setzen die Hochsprachen an, die sich gegen eine simultane Interpretation des Programmtextes wehren. Sie bevorzugen eine einmalige Übersetzung in den für den Computer verständlichen Code. Dadurch hat man nur einmal einen Zeitverlust, der durch Geschwindigkeitsgewinn belohnt wird. Man bezeichnet diese Methode als Kompilieren, wenn von einer Hochsprache über eine einmalige Übersetzung zum Prozessor-verständlichen Code gegangen wird. Dazu im krassen Gegensatz steht der Interpreter, der immer wieder neu übersetzt.

6.1 Assembler

Nachdem Sie nun mit den Tücken des Betriebssystems vertraut sind und Ihnen auch die Sprache BASIC nichts umwerfendes mehr bieten kann, ist es an der Zeit, Überlegungen darüber anzustellen, ob man nicht noch schnellere und bessere Programme schreiben kann. Wie Sie vielleicht wissen, kann man ein BASIC-Programm compilieren. Dabei wird aus dem BASIC-Text ein direkt ladbares Programm erzeugt, das auf den BASIC-Interpreter nicht mehr angewiesen ist. BASIC ist nämlich eine Interpreter-Sprache. Mit anderen Worten, der Prozessor des Computers kann mit den Befehlen von BASIC nichts, aber auch gar nichts anfangen. Aber wieso funktionieren dann die Programme?

Wenn Sie glaubten, ein BASIC-Programm gestartet zu haben, haben Sie in Wirklichkeit eine Routine in AmigaBASIC aufgerufen, die den Text Ihres Programms las, und andere kleine Programme aufrief, die die Funktionen Ihrer Befehle ausführten. Diese Programme sind in Maschinensprache geschrieben. Aha! - Aber deshalb können Sie noch lange nicht in Maschinensprache programmieren. Oder haben Sie etwa ein Programm schon mal über langweilige Zahlenketten eingegeben? Nein? Wen wundert's. Dieses Verfahren ist auch viel zu umständlich.

Man entwickelte daher einen Zwischenweg: Für die Zahlen, die einen Maschinenbefehl darstellen, kann man ein kurzes Wort verwenden, das einprägsamer ist, als eine nichts aussagende Zahl. Gibt man ein Programm mit

diesen Worten ein, programmiert man in Assembler. Für die Eingabe verwendet man einen Editor, z.B. Ed.

Bevor wir uns allerdings darüber Gedanken machen, wie wir ein Programm schreiben und eingeben, müssen wir noch klären, wie wir den Amiga dazu bewegen können, unser Programm auszuführen. Dazu benötigen wir einen Übersetzer, der den Programmtext in ausführbare Anweisungen umwandelt. Einen solchen Übersetzer nennt man Assembler - genau wie die Sprache. Für den Amiga gibt es verschiedene Assembler.

Sehr verbreitet ist z.B. der K-Seka-Assembler der Firma Kuma, der einen internen Editor besitzt. Dieser Editor ist sehr langsam - noch langsamer als der Ed. Das haut einen vom Hocker - nicht? Ein weiterer Assembler ist Assem aus dem Commodore-Entwicklungspaket. Assem arbeitet disk-orientiert, d.h. er holt sich den Quellcode von der Diskette und schreibt den Zielcode auch auf eine solche.

Für die meisten Anwendungen ist jedoch der Profimat am besten geeignet. Er erleichtert die Arbeit dadurch, daß viele seiner Funktionen mit der Maus aufgerufen werden können. Weiterhin beinhaltet er einen Editor, der bei kurzen und mittellangen Programmen sehr schnell arbeitet. Auch verfügt Profimat über viele Sonderfunktionen. Interessant ist auch die Möglichkeit, eine nur durch die Größe des Speichers begrenzte Zahl von Editor-Fenstern zu öffnen - ein System, von dem sich so manche Textverarbeitung eine Scheibe abschneiden könnte. Möchte man einen der Quelltexte assemblieren, muß der Quellcode nicht auf Diskette vorhanden sein. Der Assembler holt ihn aus dem Editor. Seinen ausführbaren Zielcode legt er im Speicher ab. Selbst bei sehr langen Programmen, erzielt Profimat enorme Geschwindigkeiten, solange sich der komplette Quellcode im Speicher befindet. Sollte während der Übersetzung ein Fehler auftreten, kann man diesen unter Umständen korrigieren, ohne den Assembliervorgang abbrechen zu müssen. Dies geht natürlich nur, wenn der Fehler auch direkt in der Zeile liegt, über die sich Profimat beschwert. Schließlich kann man mögliche Laufzeitfehler mit dem integrierten Debugger finden. Aufgrund all dieser Vorteile werden wir bei unseren Beispielen den Profimat verwenden. Nicht zuletzt deshalb, weil er mit einem Preis von 99 DM zu den preiswertesten gehört. Das mitgelieferte Handbuch enthält übrigens noch viele Anweisungen, die wir in diesem Buch nicht ansprechen können.

Bevor wir jedoch mit den ersten schweren Schritten beginnen, möchte ich Ihnen noch ein weiteres Hilfsmittel vorstellen: Einen Monitor. Lachen Sie nicht, ich meine nämlich nicht das Gerät, das Sie hoffentlich vor sich stehen haben, sondern ein Programm, das den Speicher in verschiedenen Formen anzeigen und verändern kann. Mit einem solchen Programm konnte man auf Rechnern, die nicht mit Multitasking ausgerüstet waren, sogar ganze Maschinenprogramme eingeben und abspeichern. Auf dem Amiga ist dies unter normalen Bedingungen nicht möglich, da ein Programm immer an eine andere Adresse geladen wird und das ausführbare Programm-File etliche Zusatzinformationen enthalten muß.

Um bestehende Programme zu testen und den Geheimnissen des Betriebssystems auf die Schliche zu kommen, ist ein Maschinensprache-Monitor jedoch zu gebrauchen. Eine besondere Form des Monitors ist der Debugger. Für die Arbeit in diesem Buch werden wir uns an den verbreiteten C-Monitor halten, der zwar für einen Monitor recht teuer ist, dafür aber zuverlässig arbeitet und relativ viele Funktionen bietet. Sollte Ihnen die Anschaffung des C-Monitors zu teuer sein, können Sie für viele Arbeiten auch den Debugger des Profimat verwenden. Er ist zum reinen Debuggen (= Fehler suchen) aber eher geeignet.

6.1.1 Einsteigen

Assembler arbeitet eng mit der Hardware und den Bedingungen des Computers zusammen. Wollen wir also Assembler lernen, bleibt es uns auch nicht erspart, uns mit diesen Dingen zu beschäftigen.

Der Chef des Amiga ist der M68000-Prozessor. Er koordiniert die Aktivitäten seiner Mitarbeiter und führt die Denkarbeit aus. Er ist es auch, der für die Abarbeitung der Assembler- bzw. Maschinenprogramme zuständig ist. Der von Motorola gebaute Prozessor arbeitet intern mit bis zu 32 Bit breiten Daten.

Bevor wir beginnen, ein Programm in Assembler zu schreiben, müssen wir uns noch ein paar Gedanken darüber machen, wie ein Programm intern aufgerufen, gestartet und beendet wird. Dies ist prinzipiell recht einfach. Wir wollen hier auch nicht auf die Einzelheiten eingehen. Das Programm wird geladen - wohin es geladen wird, hängt davon ab, wo wieviel Speicher frei ist und in welchen Speicherbereich es laufen soll. Dann wird in das Programm verzweigt. Um dieses wieder verlassen zu können, erfolgt dessen Aufruf als Un-

terprogramm, so daß man es durch einen Befehl beenden kann, der dem BASIC-Befehl Return gleichkommt. Dieser Befehl trägt den Namen RTS.

Nun, wo wir den ersten Befehl kennen, versuchen wir, diesen in den Assembler einzugeben, ihn zu übersetzen und auszuführen. Das Resultat dürfte sich in Grenzen halten, da ein Programm, das nur aus dem RTS-Befehl besteht, keine Wirkung hat.

Sollte dennoch etwas Besonderes passieren, ist dies bestimmt eine Fehlermeldung oder der Guru persönlich. Mit Guru ist hier übrigens eine rot blinkende Meldung am oberen Rand der Screens gemeint, die immer dann in Erscheinung tritt, wenn der Amiga abgestürtzt ist oder kurz davor steht. Er stürzt natürlich nicht richtig ab. Vielmehr trifft er auf einen Maschinenbefehl, den er nicht ausführen kann.

Ein gewöhnlicher Prozessor würde nun einfach aufhören zu arbeiten. Doch nicht der 68000er - er ruft eine Routine im Betriebssystem auf, der er genauestens mitteilt, was passiert ist. Diese Routine gibt dann den Guru aus. Sollten Sie übrigens einen zweiten Amiga, einen PC oder einen anderen Computer besitzen, der über eine RS232-Schnittstelle verfügt, können Sie über ein DFÜ-Programm und ein Nullmodem mit einem Hilfsprogramm im Amiga in Verbindung treten und den Fehler aufspüren, der für den Guru verantwortlich ist.

Dieses Hilfsprogramm ist nichts anderes als ein mittelmäßiger Monitor, der jedoch nicht über die Amiga-Tastatur sondern über die serielle Schnittstelle angesteuert wird. Aufrufen können Sie ihn, indem Sie die Guru-Meldung nicht mit der linken, sondern mit der rechten Maustaste bestätigen.

Nun aber los. Bevor Sie den Profimat des erste Mal verwenden können, müssen Sie ihn von der nicht schreibgeschützten Orginal-Diskette starten. Sie werden dann aufgefordert, Ihre persönlichen Daten einzugeben. Alles, was Sie hier eintragen, wird im Programm fest eingebaut. Änderungen sind nicht möglich.

Nachdem Sie alles eingetragen und die Richtigkeit bestätigt haben, fertigen Sie bitte zwei Kopien der Profimat-Diskette mittels DiskCopy an. Eine zur Sicherheit, die andere zum Arbeiten. Haben Sie's?

Bevor wir loslegen, müssen wir noch die Arbeitsdiskette herrichten. Alle in diesem Buch aufgeführten Beispiele funktionieren mit einem Speicher von

512 KByte, wenn Sie die Arbeitsdiskette so einrichten, wie wir es vorschlagen. Machen Sie also bitte alles mit. Auch wenn Sie zu den glücklichen Besitzern eines Mega-Bytes gehören, sollten Sie sich daran halten, um sicherzustellen, daß auch wirklich alles klappt. Denn gerade, wenn man frisch in Assembler eingestiegen ist, kann man mögliche Fehlerquellen nicht ohne weiteres auffinden. Also los:

Wichtig ist erstmal Platz. Den schaffen Sie am Besten, indem Sie die Demo-Schublade und das File "Profimat:Includes/exec/pm.i" löschen. Heißt Ihre Diskette nicht Profimat, müssen Sie statt dessen den Namen einsetzen, den Sie ihr gegeben haben.

An dieser Stelle können und wollen wir keine vollständige Anleitung zum Profimat liefern. Dies können wir allein aus Platzgründen nicht tun. Das Handbuch zu Profimat, das immerhin 140 Seiten stark ist, klärt jedoch den Großteil der aufkommenden Fragen. Wir wollen Ihnen statt dessen die Möglichkeit geben, allmählich in Assembler einzusteigen. Mit einem Handbuch zum Assembler und einem Prozessor-Buch zum 68000er kommt man als Maschinensprache-Einsteiger meistens nicht zurecht.

Wenn Sie Funktionen aus den Menüs aufrufen wollen, denken Sie bitte daran, daß immer das Fenster, dessen Menü man aufrufen möchte aktiv sein muß. Auch der Editor funktioniert nur, wenn er aktiviert ist. Das jeweils aktive Fenster kann man daran erkennen, daß sein Name im Vergleich zu den anderen Fenstern gut lesbar ist. Möchte man ein Fenster aktivieren, muß man es lediglich mit der Maus anklicken.

Wenn Sie nur 512 KByte haben, rufen Sie Profimat bitte immer auf, indem Sie zuerst die Workbench booten, dann die Profimat-Diskette einlegen, und schließlich den Assembler über sein Icon aufrufen. Verwenden Sie bitte eine Workbench-Diskette, die der Orginal-Diskette einigermaßen ähnlich ist, d.h. die also keine CLI-Befehle in die RAM-Disk kopiert oder andere Operationen ausführt, die den freien Speicher reduzieren.

Aktivieren Sie das Editor-Fenster, erscheint ein Cursor, der angibt, an welcher Stelle die nächste Eingabe angezeigt wird. Diesen Cursor können Sie mit den vier Cursortasten steuern. Bevor wir nun etwas eingeben, wählen Sie noch nacheinander im Editor-Menü "10er Block->Steuertasten", "Schreibmodus-> überschreiben" und "Speicher" an. Nach dem Aufruf von "Speicher" werden Sie gefragt, weiviel Speicher Sie dem Editor gönnen möchten. Hier reichen

10001 Byte vollkommen aus. Bestätigen Sie mit OK. Nun rufen Sie "Voreinst. sichern" aus dem Datei-Menü auf. Die Änderungen werden auf die eingelegte Assembler-Diskette gespeichert. Wenn Sie die Fenster anders ordnen, können Sie deren neue Position übrigens auch mit diesem Menüpunkt abspeichern.

Ach ja richtig. Prüfen Sie bitte, ob in den Datei-Menüs von Assembler und Editor der Menüpunkt Icon mit einem Häkchen versehen ist. Sollte dies nicht der Fall sein, sorgen Sie bitte dafür. Wie man unschwer vermuten kann, entscheidet das darüber, ob die abgespeicherten Dateien und Quelltexte mit einem Icon versehen werden oder nicht.

Und noch etwas sollten Sie wissen. Profimat erwartet von Ihnen ein paar Hilfen. Im Editor müssen alle Definitionen und Label in der ersten Spalte stehen. Diese Regel sollten Sie sich unbedingt merken, auch wenn Sie noch nicht wissen, was Definitionen und Label sind. Sie werden im weiteren Verlauf des Assembler-Kapitels jedoch noch vorgestellt. Alle Befehle dürfen nicht in der ersten Spalte beginnen. Sie müssen mindestens ein Zeichen eingerückt werden. Dies gilt sowohl für Assembler- als auch für Profimat-Befehle. An diesen Umstand werden Sie sich schnell gewöhnen. Nicht zuletzt dadurch ist der Assembler so schnell. Weiterhin müssen Sie noch wissen, daß jedes Programm mit dem Befehl END zu beenden ist. END muß, da es sich um einen Befehl handelt, mindestens ein Zeichen eingerückt werden. END ist übrigens kein Assemler- sondern ein Profimat-Befehl. Er taucht im Ziel-Code nicht mehr auf. Aber fangen wir an: Zum Einstieg geben Sie bitte den folgenden kleinen Text ein. Achten Sie darauf, das, was eingerückt ist, auch einzurücken.

```
Programm:
 rts
 end
```

Nun haben wir immerhin schon ein Programm. Es hat zwar noch keine Funktion, für die Demonstation reicht es aber vorerst aus. Ein Assembler-Programm ist erst lauffähig, wenn man es in Maschinensprache übersetzt. Beim Amiga reicht das noch nicht ganz aus. Zusätzlich werden mit dem Programm noch einige Daten abgespeichert. Diese Daten werden vom Assembler autmatisch errechnet, wenn im Ausgabe-Menü des Assemblers "normaler Code" mit einem Häkchen versehen ist. Um das Programm zu übersetzen, aktivieren wir das Assembler-Fenster. In einem Menü, das bezeichnender Weise Assembler heißt, finden Sie ganz oben den Menüpunkt "Assemblieren". Wählen Sie ihn

bitte an. Nun erscheint ein großer Requester, in dem Sie einstellen können, in welchem Speicherbereich Ihr Programm später laufen soll und darf. Dabei wird zwischen Code, Data und Bss unterschieden. Am Ende des Assembler-Kapitels werden wir auf diese drei Funktionen noch eingehen. Jetzt sind sie jedoch noch zu verwirrend. Bekannt sind Ihnen die Begriffe Chip und Fast. Sie geben an, in welchen Speicherbereich das Programm geladen werden soll. Dabei gibt es drei Möglichkeiten.

1. Wenn vorhanden, wird der Programmteil ins Fast-RAM geladen. Sollte kein Fast-RAM zur Verfügung stehen, wird stillschweigend das Chip-RAM verwendet. Dieses Verfahren stellen Sie ein, indem Sie weder Chip noch Fast anwählen.

2. Der Programmteil wird ins Chip-RAM geladen. Ist nicht genügend Chip-RAM da, bricht die Laderoutine mit einer Fehlermeldung ab. Will man diesen Modus einschalten, muß man auf Chip klicken. Ein weiterer Mausklick schaltet wieder in den normalen Modus. (1)

3. Die dritte Möglichkeit sollte man eigentlich nie verwenden. Wählt man sie durch Aktivierung von Fast an, wird der Teil, für den dieser Modus gewählt wird, ins Fast-RAM geladen. Ist nicht genügend Fast-RAM da oder startet man das Programm auf einem System ohne Speichererweiterung, wird das Programm nicht geladen. Man sollte diesen Modus wirklich nur in wenigen Ausnahmefällen verwenden.

Für unser Beispiel empfiehlt es sich, dreimal den ersten Modus zu verwenden. Dazu muß an der Einstellung nichts geändert werden. Klicken Sie auf OK, ertönt ein Ton, der signalisiert, daß der Quelltext fertig übersetzt wurde. Aktivieren Sie bitte den Editor. Im Datei-Menü finden Sie "Abspeichern". Wählen Sie diesen Menüpunkt an. In der nun erscheinenden File-Request-Box geben Sie den Programm-Namen ein (z.B. DF0:Rts.ASM).

Zur Bestätigung klicken Sie OK an. Es ist ratsam, für alle Assembler-Programme die Endung ".ASM" zu verwenden. Verwenden Sie keine Endung, erhalten die ausführbaren Programme den Zusatz ".PRG". Verwendet man bei den Quelldateien eine Endung, fällt diese am Zielfile weg. Nun haben Sie das Quellfile abgespeichert. Um auch noch das ausführbare Zielfile auf die Diskette zu bringen, müssen Sie den Assembler aktivieren und in dessen Datei-Menü wieder "Abspeichern" anwählen. Diesmal wird keine File-Request-Box

angezeigt, da der Name schon festgelegt wurde. Wollen Sie einen anderen Namen verwenden, können Sie "Abspeichern als" aufrufen.

Unser erstes Assembler-Programm haben wir eingegeben, assembliert und abgespeichert. Nun wollen wir es zur Probe aufrufen. Dazu öffen wir die Profimat- oder Daten-Diskette über das zugehörige Icon. Im Fenster der Diskette sollten nun zwei neue Bildchen zu sehen sein. Ein Textblatt und ein Rennwagen. Bei Ihnen sind diese Icons nicht vorhanden? Dafür kann es eine einfache Erklärung geben: Möglich ist, daß das Fenster der Diskette, auf die das Programm gespeichert wurde, schon geöffnet war, bevor Sie den Quelltext und das Programm darauf geschrieben haben. In diesem Fall müssen Sie das Fenster schließen und neu öffnen. Die Icons müßten nun erscheinen. Vielleicht haben Sie auch vergessen, das Abspeichern der Icons über das Menü einzustellen. Dies geht übrigens auch nur, wenn die Bildchen PROFIMAT.PRG und PROFIMAT.TXT verfügbar sind. Wenn die Icons Ihres gesicherten Programms erst einmal im Fenster zu sehen sind, brauchen Sie nicht nach jedem Abspeichern das Fesnter zu schließen und zu öffnen. Sie können das Programm dann gleich mit einem Doppelklick starten. Das wollen wir jetzt auch tun.

Führen Sie einen Doppelklick der linken Maustaste über dem Rennwagen aus. Wenn sich gar nichts tut, ist alles in Ordnung. Unser kleines Einstiegsprogramm führt ja auch noch nichts aus. Das soll sich aber ändern.

6.1.2 Grundzüge der Sprache

Um mit Assembler leichter fertig zu werden, ist es von Vorteil, BASIC zu beherrschen, um die prinzipiellen Programmiertechniken zu kennen. Wenn Sie es noch nicht getan haben, arbeiten Sie bitte das BASIC-Kapitel dieses Buches durch, da dort wichtige Grundelemente vorgestellt werden, die auch in Assembler wichtig sind.

Solange Sie in BASIC programmierten, konnten Sie einen Wert einfach in einer Variablen ablegen, der Sie einen Namen geben konnten. Wo der Wert tatsächlich gespeichert wurde, hat Sie meistens nicht interessiert. In Assembler können Sie nicht einfach sagen

```
A = 10
```

Sie müssen dazu zuerst einmal festlegen, was A eigentlich sein soll. Hier ein Beispiel:

```
Programm:
 move.l #10,A
 rts

A: ds.l 1

 end
```

Um den Überblick zu behalten, gehen wir dieses kleine Programm einmal durch.

```
Programm:
```

In der ersten Zeile haben wir eine überflüssige Sprungmarke eingesetzt, die den Namen Programm trägt. Sie kann, ähnlich wie in BASIC, mit einem Befehl angesprungen werden. Eine Sprungmarke bezeichnet man auch gerne als Label.

```
 move.l #10,A
```

MOVE ist der wichtigste aller Maschinensprachebefehle, die der 68000er versteht. Mit ihm kann man Werte und Daten verschieben. In BASIC verwendet man dafür das Gleichheitszeichen. Dabei wird jedoch die Reihenfolge von Quelle und Ziel vertauscht. In BASIC nennt man zuerst das Ziel (z.B. A) und dann der Wert (z.B. 10). In Assembler dreht man das ganze einfach um: zuerst den Wert - oder, um wieder ein neues Fachwort einzuführen, der Quelloperand - (#10) und dann das Ziel (A). Direkt hinter MOVE steht noch eine Erweiterung (.L). Der Prozessor kann nämlich mit drei verschiedenen Zahlengrößen rechnen. Neben den Bytes (.B) und Worten (.W) gibt es noch Langworte (.L). Ein Wort besteht aus zwei Byte; ein Langwort aus zwei Worten - also vier Bytes. In BASIC kann man zwar nicht mit Bytes aber immerhin mit Worten und und Langworten rechnen. Worte erhalten den Anhang "%" und Langworte "&". Variablen ohne Anhang sind sogenannte Fließkommavariablen. Darunter versteht man Zahlen, die auch Potenzen und Nachkommastellen beinhalten können. Eigentlich hätten wir also

```
A& = 10
```

schreiben müssen, um einen Befehl zu haben, der

```
MOVE.L #10,A
```

möglichst ähnlich ist.

In unserem Assembler-Beispiel müssen wir A festlegen, bevor wir etwas hineinschreiben können. Dies haben wir mit

```
A: DS.L 1
```

bewerkstelligt.

```
DS.x n
```

ist ein Profimat-Befehl, der n mal Platz für eine Variable der Größe x reserviert. Wir haben uns ein Langwort beschafft. Fällt Ihnen etwas auf? Der Speicher für A wird hinter der Stelle belegt, an der A das erste (und einzige) Mal benutzt wird. Da wir mit einem 2-Paß-Assembler arbeiten, können wir das ohne weiteres tun. Im ersten Durchgang sieht er nach, was wo benutzt und definiert wird, im zweiten bastelt er dann ein lauffähiges Programm daraus zusammen.

Vor den DS-Befehl haben wir ein Label gestellt. Labels werden meist durch einen Doppelpunkt beendet. Aber hatten wir nicht vorhin gesagt, ein Label sei eine Sprungmarke? Und nun soll man es plötzlich auch für Variablen verwenden können? Da stimmt doch was nicht! Diese Ansicht ist, wenn man von BASIC aus argumentiert, richtig. In Assembler verschwimmen die Konturen etwas. Man kann Sprungmarken, Variablen und Felder nicht mehr unterscheiden. All diese Dinge werden intern als Adressen verwaltet.

Überlegen wir uns einmal, warum das so ist: Der Prozessor holt sich den Befehl, den er auszuführen hat aus dem Speicher - soviel steht fest. Da der Speicher aber groß ist, muß er genau wissen, wo er den Befehl herholen soll. Er numeriert daher alle Bytes durch. Jedes Byte erhält also eine Adresse. Schreibt er nun Daten in den Speicher (siehe Beispiel), muß er auch wieder eine Adresse haben, an der er seinen Wert ablegen kann - die Adresse der von uns erfundenen Variablen A. Für ihn ist A also keine Variable, sondern eine Adresse von vielen im Speicher. Hätten wir in unserem Prograämmchen

```
move.l 10,A
```

geschrieben, hätte er die Zahl zehn auch als Adresse ausgelegt und deren Inhalt bei Adresse A abgelegt. Wohlgemerkt, den Inhalt. Aber woher erfährt er die Adresse von A? Die haben wir ihm doch gar nicht mitgeteilt. Richtig. Das brauchen wir auch nicht, da wir dafür unseren Hilfsarbeiter - den Assembler - haben. Im Amiga sollte man ohnehin nie auf absolute Adressen (z.B. 10) zugreifen, da man wegen des Multitaskings mächtig Ärger bekommen kann, wenn ein anderes Programm den gleichen Speicherbereich verwendet.

Maschinensprache ist sehr schnell. Maschinensprache ist aber nicht gleich Maschinensprache. So ist z.B. der Code (das Programm), den ein Compiler erzeugt hat, (fast) nie so schnell wie ein gut programmiertes reines Assemblerprogramm. Auch wenn man man sein Programm in Assembler schreibt, kann man durch spezielle Tricks die Geschwindigkeit noch weiter steigern. Nicht unbedingt als Trick kann man die Verwendung von Registern bezeichnen. Oft kommt man ohne sie nicht aus, oft sind sie aber einfach nur schneller. Die Register bilden eine besondere Form des Speichers. Sie befinden sich im Prozessor selbst und können daher enorm schnell angesprochen werden. Der 68000er gehört zu einer Prozessorgruppe, die über eine relativ hohe Register-Anzahl verfügt. Er hat 8 Daten und 8 Adreßregister. Eins dieser Adreßregister kann man jedoch nicht ohne weiteres verwenden, da es vom Prozessor benötigt wird. Daten- und Adreßregister unterscheiden sich in der Art der Anwendung. Mal darf man nur die eine, mal nur die andere Art einsetzen. Oft sind aber auch beide erlaubt. Jedes Register hat eine Breite von vier Bytes, also einem Langwort. Man kann Datenregister aber auch als Bytes oder Worte ansprechen. Bei Adreßregistern ist dies eigentlich nicht vorgesehen. Greift man auf einen Wert oft zu (z.B. den Zähler einer Schleife), sollte man diesen Wert in einem Reigster ablegen. Dadurch steigert sich die Geschwindigkeit der Abarbeitung ganz erheblich.

Jedes Register hat einen Namen. Die Daten- und Adreßregister haben auch noch eine Nummer. Folgende Register stehen zur Verfügung:

```
Datenregister: D0, D1, D2, D3, D4, D5, D6, D7
Adreßregister: A0, A1, A2, A3, A4, A5, A6, {A7}

Stack-Pointer:   SP=A7, USP, SSP

Statusregister: SR

Programmcounter: PC
```

Hier ist vermutlich eine kleine Erklärung angebracht. A7 ist das Register, das der Prozessor für seine eigenen Zwecke verwendet. Man kann es auch über die Abkürzung SP ansprechen. Mit dem Stack-Pointer werden wir später noch arbeiten. Über ihn kann man Werte schnell zwischenspeichern, ohne eine bestimme Adresse vorgesehen zu haben. USP und SSP sind auch Stack-Pointer. Mal enthält SP den Wert von USP, mal den von SSP. Dies liegt daran, daß der MC68000 in zwei Betriebsarten laufen kann. In einem normalen Modus, in dem einige Befehle nicht verfügbar sind, und in einem sog. Supervisor-Modus, in dem man besondere Operationen ausführen kann. Wir werden in diesem Buch nur den normalen User-Modus verwenden, mit dem man auch meistens auskommt. Das SR-Register ist nötig, um Zahlen zu vergleichen und bedingte Sprünge auszuführen. (IF THEN ELSE würde man in BASIC sagen.) Auch enthält es einige Informationen darüber, wie das laufende Programm abgearbeitet wird und welche Geräte den Prozessor bei seiner Arbeit stören dürfen. Zuletzt gibt es noch den Programmzähler. Dieses Register enthält die Adresse, aus der der nächste Befehl gelesen wird.

Prinzipiell legt der 68000er fast alles als Adressen aus. Neben einigen Ausnahmen können Sie ihn nur durch ein vorgestelltes "#" dazu überreden, den angegebenen Wert als Zahl zu interpretieren.

Möchten Sie eine Zahl innerhalb eines Programms mehrmals verwenden, ist es nicht ratsam diese jedesmal direkt anzugeben. Sollte es nämlich notwendig werden, die Zahl in einer anderen Programmversion zu ändern, müßten Sie sie mehrmals ändern. Auch eine Suchfunktion kann hier manchmal Fehler machen. Wollen Sie z.B. die Zahl 10 in die Zahl 20 umwandeln, weil Sie (nur als Beispiel) nicht 10 sondern 20 Zeilen darstellen wollen, ändert die Suchfunktion eventuell auch 1050 in 2050 um. Manche Suchroutinen machen diesen Fehler zwar nicht, aber eine Funktion, die so intelligent ist, zu unterscheiden, ob die im Programm stehende Zahl 10 nun die Zahl der Zeilen oder Ihre Hausnummer darstellen soll, gibt es noch nicht. Ändern Sie also die Anzahl der Textzeilen, die das Programm ausgeben soll, müssen Sie auch umziehen, damit wieder alles stimmt. Doch Spaß beiseite. Es geht natürlich auch einfacher. Sie müssen lediglich alle Werte am Anfang definieren. So können Sie sie später leicht ändern und wissen auch immer, wenn Sie im Programm einen Befehl sehen, was dieser zu bedeuten hat.

```
Zeilen=10
Hausnummer=10
```

Im Programm könnte nun die Zeile

```
move.w #Hausnummer,XXX
```

stehen. Ändern Sie

```
Zeilen=10
```

in

```
Zeilen=20
```

um, würde sich die Hausnummer nicht ändern.

Übrigens kann Ihr Assembler auch mit Binär- und Hexadezimalzahlen rechnen. Binärzahlen muß ein "%" vorangestellt werden, hexadezimale Werte werden durch "$" gekennzeichnet.

Auf den letzten Seiten haben wir Sie mit Daten nur so überschüttet. Um all das, was Sie gelesen haben, besser zu verstehen, haben wir ein kleines Programm für Sie vorbereitet, daß wir mit dem Debugger untersuchen werden. Dabei können Sie dann genau sehen, wie jeder einzelne Befehl wirkt. Doch sehen Sie selbst:

```
; ************************************************
; *                                              *
; *  Programm: MoveTest.ASM                      *
; *                                              *
; *  Mirko Langlotz                              *
; *                                              *
; *  29.08.1988                                  *
; *                                              *
; ************************************************

Anfang:                                 ; Label
 move.l     #0,d0
 move.l     #$12345678,d0
 move.w     #0,d0
 move.b     #-1,d0                      ; negatives Byte
 move.b     #11,d0
 move.w     #-2,d0                      ; negatives Wort
 move.l     d0,Mem                      ; Register->Speicher
 move.l     Mem,d2                      ; Speicher->Register
 move.l     #Mem,d3                     ; Adresse->Register
```

```
 move.l     #Mem,a0                    ;    "        "
 lea        Mem,a1                     ; ein neuer Befehl
 rts                                   ; Ende ...

Mem:                                   ; Label
 ds.l       1                          ; 1 Langwort

 end
```

Nachdem das Programm assembliert und abgespeichert ist, klicken wir den Debugger nach vorn. Bevor wir loslegen, wollen wir noch einige Einstellungen überprüfen. Sollte im Parameter-Menü "Register anzeigen" nicht abgehakt sein, ändern wir dies. Für die Zahlenausgabe wählen wir im Parameter-Pulldown "Anzeigen->HEXadezimal". Auch beim Anzeigeformat sind wir wählerisch und nehmen deshalb "Anzeigen->symbolisch".

Nun laden wir das zu testende Programm. Im Debugger-Menü gibt es dazu den Menüpunkt "Laden".

Laden Sie unser Demonstrationsprogramm MoveTest. Wenn alles geklappt hat, rufen Sie "Variablen berechnen" aus dem Debugger-Menü auf. Nach ein paar Sekunden geschäftigen Tuns, sehen wir zu unserer großen Freude die Labels im Debugger-Fenster auftauchen, die wir im Programm verwendet haben. Das Fenster kann man grob in zwei Teile aufspalten. Auf der rechten Seite werden das Programm oder dessen Daten angezeigt und auf der linken eine Liste der Register. Die erste Zeile des Programms, vor der auch das Label ANFANG steht, ist von einem Balken umgeben. Dieser Balken symbolisiert, daß es sich hier um den nächsten Befehl handelt, der beim Abarbeiten des Programms ausgeführt wird. Anscheinend scheint der Prozessor still zu stehen. In der Tat steht das Programm auch. Der Debugger kann dies mittels eines kleinen Tricks erreichen.

Was uns interessieren soll, sind die verschiedenen Möglichkeiten, das Programm Schritt für Schritt abzuarbeiten. Der Debugger zeigt dabei ständig an, was sich an den Registern und im Speicher ändert. Auch können wir uns Speicherbereiche ansehen, in denen bestimmte Daten liegen. Wie einfach das alles zu bewerkstelligen ist, wird die Praxis zeigen.

Durch "Variablen berechnen" haben wir erreicht, daß die Adressen der Labels, die wir in dem zuletzt assemblierten Programm verwendet haben, an die die Adresse angepaßt werden, an die der Debugger sein Programm geladen hat.

Wollen wir diesen Luxus verwenden, müssen wir also immer erst assemblieren und dann debuggen - auch, wenn wir seit der letzten Assemblierung nichts am Quellcode geändert haben.

Denn erst dadurch wird es möglich, zusätzlich zu den nichts sagenden Adressen, an denen die Programmteile liegen, die Labels sichtbar zu machen.

Der erste Befehl lautet

```
ANFANG:MOVE.L #0,D0
```

Werfen wir einen Blick in das Menü "Steuerung". Dort sehen wir eine ganz beachtliche Zahl von Anweisungen, die wir zur Abarbeitung verwenden können.

Anweisung	Tastenkombination
Start	R-[Amiga] [S]
Start abbrechbar	R-[Amiga] [A]
Nächster Befehl	R-[Amiga] [N]
Einzelschritt	R-[Amiga] [T]
68020-Einzelschritt	R-[Amiga] [2]

Gegen wir sie der Reihe nach durch. Start würde das Programm mit normaler Geschwindigkeit weiterlaufen lassen. Hat man sich keine Möglichkeit geschaffen, wieder in das Geschehen einzugreifen, arbeitet das Programm nun so lange selbständig weiter, bis es sich selbst beendet - oder abstürzt.

Start abbrechbar ist Start sehr ähnlich. Nach jedem ausgeführten Befehl werden in diesem Fall die neuen Register-Inhalte angezeigt. Wie der Name schon sagt, kann man den Ablauf unterbrechen. Dazu dient die [Esc]-Taste. Man kann aber auch ein zweites Mal die rechte [Amiga]-Taste und [A] betätigen.

Weiter geht es mit dem nächsten Befehl. [Amiga] [N] führt den Befehl, auf den der Programmzähler zeigt, vollständig aus. Handelt es sich um einen Unterprogramm-Aufruf, wird das ganze Unterprogramm ausgeführt. So kann man sich das mühsame Durcharbeiten von Betriebssystem-Routinen ersparen, die man ohnehin nicht zu debuggen braucht und oft auch nicht debuggen kann.

Einzelschritt führt genau einen Befehl aus. Unterprogramme können somit auch bearbeitet werden. Überlegen Sie sich immer gut, ob Sie ein Unterprogramm ausführen wollen oder ob es sinnvoller erscheint, jeden Befehl einzeln durchzugehen.

Der 68020-Einzelschritt kann von einem Computer, der keinen 68020 hat, nur emuliert werden. Profimat arbeitet so viele Einzelschritte ab, bis er auf einen der folgenden Befehle trifft. Die Register werden nur angezeigt, wenn er einen solchen erreicht.

```
Bcc, DBcc, BSR, JSR, JMP, TRAP, RTE, RTR, RTS
(RTD beim 68010)
```

Auch wenn Sie noch nicht wissen, was diese Befehle bedeuten, brauchen Sie nicht enttäuscht zu sein. Sie können, wenn Sie die Befehle später kennen, hier nachsehen, wann der 68020-Einzelschritt abbricht.

Die Move-Befehle unseres Programms können wir alle mit Einzelschritt durchgehen. Dies werden wir jetzt auch tun. Nach jedem Befehl überprüfen wir, ob sich die Register geändert haben.

Nach dem kleinen Umweg wollen wir dort fortfahren, wo wir unterbrochen hatten. Der Progammzähler zeigt immer noch auf den ersten Befehl. Drücken Sie die rechte [Amiga]-Taste+[T] ([Amiga] [T]), wandert der Programmzähler auf den nächsten Befehl. An den Registern ändert sich anscheinend nichts. Die Ursache dafür ist, daß in D0 auch schon vor Aufruf des Befehls der Wert Null stand. Das wird sich aber gleich ändern, denn

```
MOVE.L #$12345678,D0
```

überträgt die Hexadezimalzahl $12345678 als Langwort in D0. Beachten Sie das Doppelkreuz vor der Zahl. Hätten wir es im Programm vergessen, würde der Computer versuchen, den Inhalt der Adresse $12345678 als Langwort zu lesen. Nach [Amiga] [T] ist unser Wert tatsächlich in D0 gespeichert. Bevor Sie nun den nächsten Befehl ausführen, überlegen Sie bitte einmal, was er wohl bewirkt. Er lautet

```
MOVE.W #0,D0
```

Es darf spekuliert werden! Das Ergebnis erhalten Sie, durch [Amiga] [T]. Zu unserer großen Verwunderung steht in D0 nun nicht $0 sondern $12340000.

Sollte der Amiga einen Fehler gemacht haben? Nein. Das hat er nicht. Er hat genau das getan, was er laut Anweisung tun sollte - nämlich das Wort Null in D0 abzulegen. Ein Wort entspricht zwei Bytes. Da ein Register aber vier Bytes breit ist, wird nur die Hälfte verwendet. Verwendet man bei weiteren Operationen wieder die Erweiterung .W, wird der obere Teil des Registers auch nicht verwendet.

```
D0.B =        $00
D0.W =      $0000
D0.L = $12340000
```

Weiter im Text.

```
MOVE.B #-1,D0
```

lautet der nächste Befehl. -1? Wie kann man denn mit negativen Zahlen rechnen? Eine gute Frage, auf die die Entwickler von Mikroprozessoren eine gute Antwort fanden. Lassen Sie sich doch mit [Amiga] [T] überraschen. In D0 steht nach Ausführung des Befehls das Langwort $123400FF. Wir interessieren uns aber nur für das Byte: $FF. Von einer negativen Zahl ist hier nichts mehr zu bemerken. Bevor wir uns über diesen Umstand den Kopf zerbrechen, führen wir noch

```
MOVE.B #$B,D0
```

und

```
MOVE.W #-2,D0
```

aus. Der Inhalt von D0 ändert sich zuerst in $1234000B und danach dann in $1234FFFE. Was ist passiert? Beim ersten Befehl ist alles klar. $B wurde als Byte in D0 übertragen. Doch dann kommt wieder eine Zahl mit negativem Vorzeichen. Diesmal ist es -2. Weiterhin behandeln wir diese Zahl als Wort. Tatsächlich wird aber $FFFE übertragen.

Sollte $FFFE = -2 sein? Intern ist das richtig. Man kann aber sowohl mit $FFFE als auch mit -2 rechnen.

-2 als Langwort entspräche übrigens $FFFFFFFE.

Zahl	Byte	Wort	Langwort
0	$00	$0000	$00000000
1	$01	$0001	$00000001
9	$09	$0009	$00000009
-1	$FF	$FFFF	$FFFFFFFF
-2	$FE	$FFFE	$FFFFFFFE
-3	$FD	$FFFD	$FFFFFFFD
-9	$F7	$FFF7	$FFFFFFF7
$F7	$F7	$00F7	$000000F7

Alles klar? Zieht man von $00 den Wert $01 ab, erhält man $FF. Nochmal -1, und wir sind bei $FE. Bei Addition und Subtraktion kann man ohne besonderen Aufwand mit negativen Zahlen arbeiten.

```
  $ (1) 0 0              0
 -$     0 1          -   1
  $ (0) F F             -1


  $ (1) 0 0              0
 +$     F F          +  -1
    (0) F F             -1
```

Die Zahlen der linken Rechnung wurden, um den Gedankengang zu erleichtern, um eine Zahl in Klammern ergänzt, die das Verständnis erleichtern soll.

Ein bißchen anstrengend war das ja schon. Der nächste Befehl bringt uns wieder in vertraute Gewässer.

```
MOVE.L D0,MEM
```

Das Langwort in D0 wird im Speicher ab Adresse MEM abgelegt. MEM hatten wir als Label definiert. Führen Sie den Befehl bitte aus. An den Registern ändert sich nichts. Dafür aber im Speicher. Dies können wir leicht feststellen. Dazu ist im Parameter-Menü "Anzeigen->HEX-Dump" anzuwählen. Das Programm verschwindet. Dafür kommen viele Zahl ins Fenster. Über [Amiga] [L] fordern wir nun ein Formular an, in dem wir einstellen, welchen Speicherbereich wir sehen wollen. Wir geben

```
MEM
```

ein. In der ersten Zeile sieht es nun so aus:

```
AAAAAAAA: 12 34 FF FE XX XX XX XX
```

XX interessiert uns nicht. AAAAAAAA ist die Adresse, an der MEM liegt, und 12 34 FF und FE sind die Bytes, die hier abgelegt sind. Sie ergeben unser Langwort $1234FFFE. Schalten wir mit "Anzeigen->symbolisch" wieder die normale Anzeige ein. Um das Programm wieder zu sehen, holen wir es über [Amiga] [L] wieder in die Anzeige. Wir geben

```
ANFANG
```

als Adresse an.

```
MOVE.L MEM,D2
```

ist der nächste Befehl. Erwartungsgemäß kopiert er das Langwort ab MEM in Register D2. Doch was ist das?

```
MOVE.L #MEM,D3
```

Probieren Sie es aus. In D3 müßte nun ein Wert angezeigt werden. Welcher Wert das ist, hängt davon ab, an welche Speicherstelle das Programm geladen wurde. Jedensfalls stellt er die Adresse von MEM dar. Das kann man sogar beweisen. Ein paar Zeilen weiter unten steht das Label MEM. Die Zahl davor ist die Adresse dieses Labels. Sie muß mit dem Wert, der nun in D3 steht, identisch sein. Adressen müssen übrigens immer als Langworte behandelt werden! Das Doppelkreuz lieferte uns den "Wert" von MEM. Hätten wir es weggelassen hätten wir den Inhalt erhalten. #MEM wird auch im nächsten Befehl verwendet. Zielregister ist diesmal A0. Ansonsten ist alles gleich geblieben. Alles? Fast alles. Aus dem Befehl MOVE.L, den wir im Quellcode verwendet hatten, macht der Debugger jetzt MOVEA.L!

```
MOVEA.L #MEM,A0
```

ist dort zu lesen. Eigentlich hätten wir im Quellcode auch den Zusatz "A" verwenden müssen, da wir als Ziel-Parameter ein Adreßregister angegeben haben. Profimat nimmt uns diese lästige Arbeit jedoch ab.

```
LEA MEM,A1
```

ist der nächste Befehl. LEA kennen wir noch nicht. Er übergibt die Adresse vor dem Komma dem Adreßregister hinter dem Komma. In diesem Fall hätten wir also auch

```
MOVE.L #MEM,A1
```

schreiben können. Richtig. Aber überflüssig ist LEA deshalb noch lange nicht. Man kann mit LEA nämlich auch Adressen berechnen, was mit dem Doppelkreuz und MOVE nicht in vollem Umfang geht.

An dieser Stelle noch ein kleiner Trick. Wenn Sie in einer Speichervariablen öfters nachsehen wollen, was diese enthält, müssen Sie dazu nicht jedesmal in den HEX-Modus umschalten und die Adresse eingeben. Sie können sich über den Inhalt der Variablen und/oder deren Adresse ständig informieren lassen. Wie das geht? Ganz einfach. Positionieren Sie den Mauszeiger über der Registeranzeige. Nun bringen Sie ihn über die Zeile eines Registers, das Sie nicht ständig im Auge behalten müssen. Sie können auch die Zeile nehmen, in der

```
L:4
```

steht. Befindet sich der Mauszeiger über der gewünschten Zeile, drücken Sie zweimal die linke Maustaste. Dieses Verfahren, eine Funktion aufzurufen, kennen Sie schon von der Workbench. Man nennt es Doppelklick.

Nachdem Sie den Doppelklick ausgeführt haben, wird ein Formular geöffnet, in dem Sie drei Dinge einstellen können. Erstens das, was angezeigt werden soll. Dabei können Sie auch Formeln (z.B. MEM+2) oder ähnliches verwenden. Auch verschiedene Adressierungsarten können verwendet werden. Adressierungsarten nennt man die Formate, in denen man einen Wert angeben kann. Wir kennen bereits die absolute (MEM) und die unmittelbare Adressierung (#10 oder #MEM). Weiterhin sind auch D0, A0 usw. Adressierungen. Wenn wir später noch andere Adressierungsarten kennenlernen, können wir diese in unserem Formular auch verwenden.

Vorerst geben wir ganz einfach

```
MEM
```

ein. L:4 müssen wir natürlich mit [Del] oder [Amiga] [X] entfernen. Dann ist noch zu entscheiden, ob die Darstellung ein Byte, ein Wort oder ein Langwort verwenden soll. Stellt man das darübergelegene Gadget auf "Adresse", ist dies ohnehin uninteressant. Dann wird nämlich nicht der Inhalt sondern die Adresse von MEM angezeigt. Und Adressen sind ja bekanntlich immer Langworte. Wir stellen "Inhalt" und ".L" ein. Bestätigen müssen wir das ganze mit OK. Nun wird ständig der Inhalt von MEM als Langwort angezeigt.

Vorerst haben wir genug debugged. Wir beenden das geladene Programm mit "Löschen" oder "Start".

6.1.3 Wie sag ich's ihm? - Die Adressierungsarten

Um gute Assemblerprogramme zu schreiben, muß man nicht nur die Befehle, sondern auch die Adressierungsarten kennen. Die Maschinensprache des 68000ers ist nicht deshalb so leistungsfähig, weil sie viele Befehle hat, sondern weil sie die relativ niedrige Zahl der Befehle mit vielen Adressierungen verbinden kann. Bevor wir uns daher auf weitere Befehle stürzen, nehmen wir uns erst einmal die Adressierungsarten vor. Haben Sie die Vorteile ausgefallener Adressierungen erst einmal erkannt, werden Sie schon eher glauben, daß Maschinensprache fast so einfach wie BASIC ist.

Fangen wir mit "Adreßregister indirekt" an. Wie man aus der Bezeichnung ersehen kann, hat das irgendetwas mit den Registern A0-A7 zu tun. Statt

```
MOVE.W #10,Label
```

könnte man auch

```
LEA Label,a0
MOVE.W #10,(a0)
```

schreiben. In diesem Fall ist das etwas umständlich. Bevor wir uns aber über den Sinn dieser Adressierung unterhalten, wollen wir noch deren Funktionsweise beschreiben. Der LEA-Befehl ermittelte die Adresse der Variablen Label. Nun wird der Wert (#) Zehn als Wort ab der Adresse abgelegt, auf die A0 zeigt. Da sich in A0 zur Zeit die Adresse von Label befindet, wird die Zahl in Label abgelegt. Kennzeichen für diese Adressierungsart sind die Klammern, die das Adreßregister einschließen. Nun zu Sinn und Unsinn der indirekten

Adressierung über ein Adreßregister. Im obigen Beispiel ist es in der Tat einfacher, Label absolut zu adressieren. Was aber, wenn man einen Wert in eine Speicherstelle schreiben möchte, deren Adresse sich ändert, oder die bei der Erstellung des Programms noch nicht feststeht? Dann muß man indirekt adressieren. Möchte man eine Routine schreiben, die auf eine bestimmte Struktur zugreift, kommt man auch um das indirekte Verfahren nicht herum, da man auf gleichartige Strukturen an verschiedenen Speicherstellen zugreifen muß. In solchen Fällen ist die indirekte Adressierung jedenfalls Gold wert.

Für die Arbeit mit Strukturen ist eine andere Adressierung noch viel besser geeignet. Es handelt sich um die Adressierung über Adreßregister und 16-Bit-Distanzwert. Zur Verdeutlichung wieder zwei gleichwertige Beispiele:

```
MOVE.W #100,Label+20

LEA Label,a0
MOVE.W #100,20(a0)
```

Der Distanzwert wird vor der Klammer angegeben, die vor dem Adreßregister steht. Man kann die Adressierungen natürlich auch als Quelle verwenden. Beispiel:

```
MOVE.W 20(a0),d0
MOVE.W (a0),d1
```

Die nächste Adressierung verwendet zwei Register und einen Distanzwert. Vorweg ein Beispiel.

```
LEA LABEL,A0
ADD.L D0,A0                              ; Addition
ADD.L #30,A0                             ; Addition
MOVE.W #100,(A0)

MOVE.W #100,30(A0,D0.L)
```

A0, D0 und 30 werden summiert. Das Ergebnis wird als Adresse verwendet, an die dann ein Wert geschrieben wird. Bei der neuen Adressierungsart kann man das zweite Register (hier D0) sowohl als Wort als auch als Langwort verwenden. Möchte man das ganze Langwort addieren, muß man .L an das Register anhängen. Verwendet man nur ein Wort, kann man sich den Zusatz .W sparen. Allgemein kann man .W meistens weglassen. MOVE ohne Endung

wird z.B. als MOVE.W ausgelegt. Um sicherzugehen, sollte man jedoch immer schreiben, was gemeint ist.

Nun wollen wir uns noch zwei weitere Adressierungsarten ansehen. Sie werden gerne in Verbindung mit dem Stack verwendet. Der Stack ist ein Speicherbereich, in dem Daten ohne großen Programmieraufwand gespeichert werden können. Auf diese Daten greift man nach dem LIFO-System zu. D.h. den Wert, den man zuerst ablegt, muß man auch zuerst zurückholen. Der Stack eines Programms ist, wenn nicht anders festgelegt, 4000 Bytes groß. Für Assemblerprogramme reicht das völlig aus - C-Programmen sollte man das doppelte bis dreifache gönnen. Aber wo liegt der Stack? Gute Frage - doch völlig belanglos. Die Adresse des Stacks braucht den Programmierer meist nicht zu interessieren. Er kann den Stack über das Register A7 ansprechen. Dieses Register enthält den aktuellen Stack-Pointer. Um Verwechslungen zu vermeiden, wollen wir statt A7 die Abkürzung SP verwenden.

Angenommen, wir haben in den Registern D0 und D1 je ein Wort und in D2 ein Langwort, das wir an einer späteren Stelle noch einmal brauchen. Vorher müssen wir diese Register aber für eine andere Aufgabe einsetzen. Nun könnten wir deren Inhalte einfach im Speicher sichern. Dazu müssen wir aber Platz schaffen. Viel einfacher ist es, wenn wir die Werte auf dem Stack ablegen. Später können wir sie zurückholen (Wie bei einem richtigen Stapel). Achten Sie darauf, alles, was Sie auf dem Stack deponieren, auch wieder von dort wegzunehmen, da Sie sonst das Programm nur durch einen Absturz beenden können. Der Stack wird nämlich auch für die Rücksprungadressen von Unterprogrammen verwendet. Doch zurück zu unserem Beispiel.

```
; ************************************************
; *                                              *
; *  Programm: Stack_1.ASM                       *
; *                                              *
; *  Mirko Langlotz                              *
; *                                              *
; *  30.08.1988                                  *
; *                                              *
; ************************************************

Anfang:

 MOVE.W #$10,D0
 MOVE.W #$20,D1
 MOVE.L #$30,D2
```

```
MOVE.W D0,-(SP)
MOVE.W D1,-(SP)
MOVE.L D2,-(SP)

MOVE.L #$AAA,D0
MOVE.L #$BBB,D1
MOVE.L #$CCC,D2

MOVE.L (SP)+,D2
MOVE.W (SP)+,D1
MOVE.W (SP)+,D0

RTS

END
```

Geben Sie dieses Programm ein, assemblieren Sie es, und speichern Sie es ab. Da wir die Register-Ergebnisse am Ende nicht ausgeben, müssen wir das Programm wieder mit dem Debugger steuern. Das wollen wir jetzt auch tun.

Sie müßten mittlerweile in der Lage sein, den Debugger selbst zu bedienen. Wenn nicht, lesen Sie bitte noch einmal auf den vorangegangenen Seiten nach. Haben Sie mit dem Debugger die Labels umgerechnet, können Sie das Programm auf einfache Weise mehrmals abarbeiten, indem Sie den Programmzähler immer wieder auf das Label Anfang zeigen lassen. Achten Sie, wenn möglich, darauf, immer erst bis zum RTS zu kommen, bevor Sie den Programmzähler wieder auf den Programmanfang zeigen lassen. Sonst könnte (theoretisch) der Stack zu voll werden. Außerdem würden Sie beim zweiten Durchlauf andere Ergebnisse erhalten, wenn Sie den Stack-Pointer in ein anderes Register umkopieren.

Bevor wir nun die neuen Adressierungsarten erklären, die wir in diesem Programm verwendet haben, bitten wir Sie, einmal alle Befehle (außer RTS) mit [Amiga] [T] auszuführen. Dann rufen Sie "Register ändern" über [Amiga] [R] auf. In dem nun erscheinenden Formular tragen Sie

```
PC=Anfang
```

ein. Nach der Bestätigung mit OK können Sie das Programm ein zweites Mal Schritt für Schritt debuggen. Die ersten drei Befehle tragen die Werte $10, $20 und $30 in die Register D0, D1 und D2 ein. D0 und D1 werden als Wort, D2 als Langwort verwaltet. Dann werden die drei Register auf den Stack gelegt. Dabei ist an der Registeranzeige keine Veränderung zu bemerken. Nun

werden D0-D2 mit verschiedenen Langworten gefüllt. Die ursprünglichen Werte scheinen verloren zu sein. Doch die nun folgenden Befehle zaubern die alten Inhalte wieder zurück. Dabei wird der Wert, der zuletzt auf den Stack gelegt wurde, zuerst geholt. Zuletzt geholt wird Register D0. Es wurde dafür zuerst auf den Stack befördert. Diese umgekehrte Reihenfolge muß unbedingt eingehalten werden!

Rufen Sie nun [Amiga] [S] auf, um das Programm automatisch weiterarbeiten zu lassen und damit zu beenden.

Nach dieser Vorführung des Stacks wollen wir endlich nachsehen, wie er funktioniert. Um einen Wert auf den Stack abzulegen, haben wir den Befehl

```
MOVE.? Ausdruck,-(SP)
```

verwendet. Das Fragezeichen vertritt die Datengröße. Da SP in Klammern gesetzt ist, können wir vermuten, daß es sich hier um eine indirekte Adressierungsart handelt. Das Minuszeichen vor der Klammer hat folgende Bewandnis:

Das in Klammern stehende Adreßregister (hier SP bzw. A7) wird zuerst um die Datengröße des Ausdrucks erniedrigt. Dann wird der Wert ganz normal an der Adresse abgelegt, auf die SP zeigt. Um wie viele Bytes SP verringert wird, hängt, wie schon gesagt, davon ab, ob man mit MOVE.B, MOVE.W oder MOVE.L arbeitet. Folglich muß man dann, wenn man Daten vom Stack zurückholt, immer die gleiche Datengröße verwenden, mit der man die Daten abgelegt hat.

Um die Werte zurückzuholen, haben wir

```
MOVE.? (SP)+,Ausdruck
```

eingesetzt. Hierbei wird die Adresse, auf die SP verweist, zuerst ausgelesen, bevor SP erhöht wird. Man nennt dies übrigens "Adreßregister indirekt mit Postinkrement". Die Adressierungsart -(SP) nennt man "Adreßregister indirekt mit Predekrement".

Wenn Sie sehen möchten, wie sich der Stapelzeiger ändert, können Sie unser Programm von eben so modifizieren, daß der Stapel-Zeiger immer in ein Register übertragen wird, das vom Debugger angezeigt werden kann.

```
; ************************************************
; *                                              *
; *  Programm: Stack_2.ASM                       *
; *                                              *
; *  Mirko Langlotz                              *
; *                                              *
; *  30.08.1988                                  *
; *                                              *
; ************************************************

Anfang:

 MOVE.W #$10,D0
 MOVE.W #$20,D1
 MOVE.L #$30,D2

 MOVE.L SP,A0
 MOVE.W D0,-(SP)
 MOVE.L SP,A0
 MOVE.W D1,-(SP)
 MOVE.L SP,A0
 MOVE.L D2,-(SP)
 MOVE.L SP,A0

 MOVE.L #$AAA,D0
 MOVE.L #$BBB,D1
 MOVE.L #$CCC,D2

 MOVE.L (SP)+,D2
 MOVE.L SP,A0
 MOVE.W (SP)+,D1
 MOVE.L SP,A0
 MOVE.W (SP)+,D0
 MOVE.L SP,A0

 RTS

 END
```

6.1.4 Programmierung

In Maschinensprache kann man Daten nicht nur verschieben. Man kann auch rechnen. Der 68000er kann neben einfachen Rechnungen sogar noch multiplizieren und dividieren. Addition und Subtraktion gehören ohnehin zu seinen

leichtesten Aufgaben. Allerdings wird das Ergebnis einer Rechnung immer in einem der Quelloperatoren abgelegt.

```
ADD.W D0,D1
```

addiert D0 zu D1 (D1=D1+D0). Umgekehrt funktioniert SUB:

```
SUB.W D0,D1
```

entspricht D1=D1-D0.

Die meisten Befehle liefern ein Ergebnis. Je nachdem, wie dieses Ergebnis aussieht, macht sich der Prozessor gewisse "Notizen". Diese legt er im SR-Register ab.

Das Statusregister unterteilt sich in mehrere Flags.

Flag	Bedeutung
(C) Carry	Wird immer gesetzt, wenn ein Übertrag über die Datenbreite entsteht.
(V) Overflow	Auch hier geht es um einen Übertrag. Overflow wird allerdings dann gesetzt, wenn ein Übertrag ins höchstwertige Bit erfolgt.
(Z) Zero	Ist das Ergebnis Null, setzt der Prozessor das Z-Flag.
(N) Negativ	Sollte das höchste Bit gesetzt sein, wird auch das N-Flag eingeschaltet. Rechnet man mit vorzeichenbehafteten Zahlen, kann man daran erkennen, daß der aktuelle Wert negativ ist.
(X) Extended	Dieses Flag wird von den Rechenbefehlen genauso geändert wie das Carry-Flag. Von anderen Anweisungen wird es nicht beeinflußt.

Neben den gewöhnlichen Flags liegen im SR-Register noch zwei System-Flags, die zum Debuggen und für das Betriebssystem wichtig sind. Weiterhin kann man noch wählen, welche Unterbrechungen des Prozessors durch die Hardware zulässig sind.

Schön, aber was nützt es uns, wenn der Computer bei jeder Gelegenheit die Flags verändert? Mit den Flags kann man unter anderem bedingte Sprünge ausführen. Doch nicht nur das. Auch andere Tricks sind möglich. Die Befehle zur bedingten Verzweigung müssen nicht unbedingt direkt auf die Flags bezo-

gen werden. Es ist verständlicherweise einfacher zu verstehen, wenn man sagt "verzweige, wenn Wert 2 größer oder gleich Wert 1 ist", als wenn man mit den Flags eine Bedingung formulieren müßte. Das könnte sich dann z.B. so anhören: "Verzweige, wenn N=1 und V=1 und Z=0 oder N=0 und V=1 und Z=0". Glücklicherweise bleibt Ihnen dieser Aufwand erspart.

```
CMP Wert1,Wert2     ; Vergleich
BEQ Sprungmarke     ; Springe, wenn gleich
```

Um zwei Werte miteinander zu vergleichen, steht der CMP-Befehl bereit. In Abhängigkeit davon, welcher der Werte größer ist, werden verschiedene Flags gesetzt. Das braucht uns aber nicht zu interessieren. Wir testen nur die Ergebnisse ab. Dazu verwenden wir den Befehl Bcc. Für cc kann man eine Bedingung einsetzen, die sich auf den zweiten Wert des CMP-Befehls bezieht. Lautet die Bedingung z.B. "größer", muß der zweite Wert größer gewesen sein als der erste. Beachten Sie bitte, daß Bcc meistens direkt hinter CMP folgen muß, da die Flags, über die der Vergleich abgewickelt wird, auch von anderen Befehlen beeinflußt werden. Folgende Bedingungen sind möglich:

cc	englisch	deutsch
Für Zahlen ohne Vorzeichen:		
HI	higher	größer
LO	lower	kleiner
HS	higher or same	größer oder gleich
LS	lower or same	kleiner oder gleich
Für alle Zahlen:		
EQ	equal	gleich
NE	not equal	ungleich
Für Zahlen mit Vorzeichen:		
PL	plus	positiv
MI	minus	negativ
VC	Overflow clear	kein Überlauf
VS	Overflow set	Überlauf
GT	greater than	größer
LT	less than	kleiner
GE	greater or equal	größer oder gleich
LE	less or equal	kleiner oder gleich
Alternativ:		
CC	Carry clear	HS
CS	Carry set	LO

CC und CS sollte man in Verbindung mit CMP nicht gebrauchen. Diese Bedingungen eignen sich von der Namensgebung her eher bei Befehlen, die Bits ins Carry-Flag schieben. Dazu später noch mehr.

"Eine umfangreiche Liste" kann man dazu nur sagen. Es gibt so viele Dinge, die man sich merken kann. Sie könnten z.B. nach und nach diese Bedingungscodes auswendig lernen. Auch können Sie versuchen, sich alle Parameter zu merken, mit denen man die Funktionen des Betiebssystems aufruft.

Das alles können Sie machen - Sie können es aber auch bleiben lassen, und sich die Infomationen, die Sie brauchen, von Profimat geben lassen. Er weiß zwar auch nicht alles, aber immerhin ist ihm bekannt, welche Befehle es gibt und welchen Befehl man mit welcher Adressierungsart verwenden kann und welchen nicht.

Auch kennt er alle Funktionen der Libraries mit den zugehörigen Parametern. Alle Informationen hat er in einem übersichtlichen Stichwortverzeichnis parat, das auch nachträglich noch geändert werden kann. Um diese Informationen zu nutzen, müssen Sie nun den Editor aktivieren und die Taste [Help] drücken. Nach einiger Zeit erscheint ein riesiges Fenster, über dessen Menüleiste Sie zuerst das Hauptgebiet auswählen müssen.

Dann können Sie die Informationen, die Profimat dazu auf Lager hat, durchblättern oder nach bestimmten Dingen suchen lassen. Wenn Sie fündig geworden sind. Schließen Sie das Fenster wieder und arbeiten weiter. Gut - oder?

Wollen Sie z.B. die Bedingungscodes (Condition Codes) abfragen, wählen Sie den Menüpunkt "EA". Er ist meistens schon eingestellt. Hier erfahren Sie alles über die Befehle des 68000, 68010 und 68020. Ganz oben ist ein kleines Kästchen zu sehen. Klicken Sie es an, erscheint darin ein Cursor. Löschen Sie nun bitte den Inhalt mit [Amiga] [X], [Del] oder [Backspace]. Nun geben Sie

```
CC
```

ein und drücken [Return]. Sofort erscheinen alle Bedingungscodes mit Beschreibung. Übrigens können Sie hier auch Kurzanleitungen zu Profimat-Befehlen einholen.

ADD, SUB, CMP und Bcc haben wir eben kennengelernt. Sehen wir uns ihre Wirkung mal an einem kleinen Programm an.

```
; ************************************************
; *                                              *
; *  Programm: Umwandlung_1.ASM                  *
; *                                              *
; *  Mirko Langlotz                              *
; *                                              *
; *  30.08.1988                                  *
; *                                              *
; ************************************************
```

Nach dieser informativen Einleitung kommen wir zum eigentlichen Programm. Um später besser debuggen zu können, verwenden wir auch hier wieder unser Label "Anfang". Dann bauen wir eine Schleife auf. In dieser Programmversion bedienen wir uns eines Zählregisters (D0.W), das wir zu den Adressen des Textanfangs und des Zielbereichs hinzuaddieren. Da die Schleife irgendwann beendet werden muß, legen wir fest, daß der umzuwandelnde Text mit einem Null-Byte abgeschlossen sein muß. Dadurch bleibt uns das Abzählen der Zeichen erspart. Es ist ohnehin üblich, das Ende von Strings so zu markieren. Da es bekanntlich zu jeder Regel eine Ausnahme gibt, weisen wir darauf hin, daß man die Länge eines Strings auch ohne Null-Byte feststellen kann. Er muß dann im BCPL-Format vorliegen. Sein erstes Byte gibt dann die Länge an.

Damit wir an der richtigen Stelle beginnen, setzen wir unseren Zähler auf Null. In A0 und A1 legen wir die Adressen von Quell- und Zielbereich ab.

```
Anfang:
 move.w     #0,d0          ; Zähler
 lea        TextAlt,a0     ; Adresse von alten Text
 lea        TextNeu,a1     ; Adresse des neuen Textes
```

Jetzt beginnen wir mit der eigentlichen Schleife. Um immer wieder an den Anfang springen zu können, verwenden wir wieder ein Label. Dann lesen wir ein Byte aus. Die Adresse errechnet sich aus dem Anfang des Textes (A0) und dem Zähler (D0.W). Wir verwenden dazu die bereits vorgestellte Adressierung. Den Distanzwert vor der Klammer müssen wir bei Profimat nicht angeben. Er geht dann davon aus, daß er einen Distanzwert Null zu verwenden hat. Das ausgelesene Byte legen wir in D1 ab, da D0 ja schon mit dem Zähler belegt ist.

```
Schleife:                          ; Anfang der Schleife
 move.b     (a0,d0.w),d1           ; Wert holen
```

Nun sind wir an der Stelle angekommen, an der alle Zeichen in Großschrift umgewandelt werden sollen. Der Einfachheit halber beachten wir nur die Buchstaben a-z (öüä... vergessen wir ganz dezent). Da die Buchstaben "a"-"z" in der ASCII-Tabelle direkt hintereinanderliegen, müssen wir alle Zeichen innerhalb dieses Bereichs umwandeln. Den verbleibenden Rest übernehmen wir unverändert. Zuerst sehen wir nach, ob der ASCII-Code in D1 kleiner ist als der des kleinen "a". Dabei können wir "a" in direkter Schreibweise angeben, wenn wir die Anführungszeichen verwenden. Dadurch behält man den Überblick und muß sich nicht mit Zahlenkolonnen beschäftigen. Bevor wir ausschließen können, daß unser Zeichen im Bereich von a-z liegt, müssen wir auch noch die obere Grenze abtesten.

```
 cmp.b      #"a",d1                 ; Vergleich mit "a"
 blo        Normal                  ; kleiner ?
 cmp.b      #"z",d1                 ; Vergleich mit "z"
 bhi        Normal                  ; größer
```

Jetzt wird umgewandelt. Wir ziehen dazu die Differenz Groß- und Kleinbuchstaben von D1 ab. Dazu muß man allerdings wissen, daß die Zeichen in Kleinschrift höhere Codes haben als ihre großen Partner.

```
 sub.b      #("a"-"A"),d1              ; a-z umwandeln
```

Jetzt ist die Stelle gekommen, an die wir springen, wenn das Zeichen nicht umgewandelt werden soll. Dies markieren wir mit dem Label Normal, das von den Vergleichen (s.o.) angesprungen wird. Wir speichern den Code in D1 in den Zielbereich. Als Index dient uns wieder der Zähler in D0.

```
Normal:                         ; Label für andere Zeichen
 move.b     d1,(a1,d0.w)        ; Zeichen speichern
```

Nun müssen wir die Schleife beenden. Dazu erhöhen wir zuerst den Zähler. Dann testen wir, ob es sich bei dem zuletzt übertragenen Zeichen um ein Null-Byte handelt. Ist dies nicht der Fall, springen wir wieder zum Schleifenbeginn. Ansonsten verlassen wir das Programm mit RTS.

```
 add.w      #1,d0                   ; Zähler erhöhen
 cmp.b      #0,d1                   ; Endmarkierung erreicht?
```

```
bne         Schleife          ; Nein, dann weiter
rts                           ; Ja, dann Ende
```

Bleiben nur noch die Daten. Hier wenden wir auch gleich einen neuen Befehl an: DC. Mit DC kann man Daten (in unserem Fall den Text) im Programm ablegen. DC muß mit der Kennung für die gewünschte Datengröße versehen werden. Da wir Bytes verwenden, können wir auch ganze Texte verwenden, die in Anführungszeichen eingeschlossen werden müssen. Am Schluß legen wir ein Null-Byte ab, das für das Programm das vereinbarte Ende darstellt. Wir haben dafür zwar eine eigene Zeile verwendet, hätten es aber auch an die letzte Textzeile anhängen können. Dann hätten wir ihm jedoch ein Komma voranstellen müssen, um eine Trennung vom String zu gewährleisten. Möchte man übrigens mehrere Zahlen ablegen, kann man diese auch in einer Zeile eingeben. Man muß sie wie gesagt, nun durch Kommata voneinander trennen.

```
TextAlt:                                ; alter Text
 dc.b       "Umwandlung in Großschrift. "
 dc.b       "Sonderzeichen wie $ * # _ @ ... "
 dc.b       "werden nicht veraendert."
 dc.b       0                           ; Ende
```

Den Speicher für den umgewandelten String reservieren wir mit der uns schon bekannten Anweisung DS. Da wir mit dem Speicher nicht sparen müssen. Geben wir hier 200 Bytes an. Dadurch wird das Programm 200 Bytes länger als nötig. Später werden wir jedoch zwei Verfahren kennenlernen, wie wir dieses Problem umgehen können.

```
TextNeu:                        ; Speicher für Umwandlung
 ds.b       200

 end
```

Das Programm gibt den umgewandelten String nicht aus. Sie werden wohl oder übel wieder den Debugger zum Testen nehmen müssen. Dazu noch einige Tips. Ganz oben in der linken Hälfte des Fensters werden nach jedem Einzelschritt die Flags angezeigt. Klicken Sie auf eine der Nullen oder Einsen, können Sie eine Was-Wäre-Wenn-Simulation durchführen. Stellen Sie z.B. das Z-Flag nach der Ausführung von

```
CMP.B #0,D1
```

auf Eins, wird das Programm früher als vorgesehen beendet, da der nachfolgende BNE-Befehl nicht mehr verzweigt.

Das Z-Flag wird immer dann gesetzt, wenn zwei Werte bei CMP identisch sind. Bei allen Operationen, die ein Ergebnis liefern oder auch einfach einen Wert kopieren (MOVE), wird das Z-Flag gesetzt, wenn das Ergebnis oder der Wert gleich Null sind. Der Eindruck, das Z-Flag hätte eine Doppelfunktion ist aber falsch. CMP tut nämlich nichts anderes, als den ersten Wert vom zweiten abzuziehen. Wenn dabei Null herauskommt, also Wert 1 mit Wert 2 identisch ist, wird das Zero-Flag eingeschaltet. Um zu testen, ob ein Wert Null ist, braucht man CMP übrigens gar nicht. Man muß mit dem Wert nur irgend etwas tun. Es reicht aus, ihn in ein anderes Register zu kopieren. Man kann also nach jedem MOVE-Befehl mit BEQ und BNE einen bedingten Sprung ausführen. Gleiches gilt auch für das N-Flag und die dafür zuständigen Befehle BVC und BVS.

```
MOVE.W Var1,Var2
BEQ Sprungmarke    ; Springe, wenn Var1 (und Var2) =0
```

Nun kann aber folgender Fall eintreten: Ein Wert wird in D0 eingelesen. In Abhängigkeit davon, ob dieser Wert Null oder nicht Null ist, möchte man etwas später einen bedingten Sprung ausführen. Zuvor müssen aber noch andere Dinge erledigt werden, die die Flags beeinflussen. Um trotzdem richtig verzweigen zu können, muß man einen neuen Befehl verwenden: TST. TST setzt und löscht Z- und N-Flag in Abhängigkeit vom adressierten Wert. Hier ein praktisches Beispiel:

```
MOVE.W Adresse,D0  ;entscheidender Wert
MOVE.W #10,D1  ;Z-Flag wird unbeabsichtig verändert.
TST.W D0           ;D0=0 ?
BEQ ...            ;Verzweige, wenn ja
```

In der zweiten Version unseres Buchstaben-Umwandlungsprogramms haben wir den TST-Befehl übrigens auch eingesetzt. Man kann ihn dort aber auch weglassen, da zuvor D1 bewegt wird. Ansonsten haben wir das Programm dadurch optimiert, daß wir den Zähler der Schleife weggelassen haben. Da wir aber irgend etwas erhöhen müssen, erhöhen wir gleich die Adreßregister. Und wo liegt da der Vorteil? Nun wir können dazu die Postinkrement-Adressierung verwenden. Sehen Sie es sich doch mal unverbindlich an. Wenn es Ihnen zu langweilig wird, die Schleife etliche Male im Einzelschritt durchzugehen, können Sie einen automatischen Breakpoint verwenden. Dazu müssen Sie die

linke Maustaste über der Zeile drücken, auf die der Programmzähler (kenntlich durch den Balken) zeigt. Nun steuern Sie mit noch immer gedrückter Maustaste die Zeile an, bis zu der das Progamm weiterlaufen soll. Dort geben Sie die Taste frei. Der Mauscursor nimmt nun wieder seine normale Form an. Dann startet der Debugger das Programm. Kommt es an die markierte Stelle, bricht es ab. Diese Technik, einen Breakpoint (deutsch: Unterbrechungspunkt) zu setzen, sollten Sie einmal kennengelernt haben. Sie ist in mittelgroßen Programmen sehr nützlich. Bei unserem kleinen Testobjekt ist sie höchstens eine Spielerei.

```
; ************************************************
; *                                              *
; *  Programm: Umwandlung_2.ASM                  *
; *                                              *
; *  Mirko Langlotz                              *
; *                                              *
; *  30.08.1988                                  *
; *                                              *
; ************************************************

Anfang:
 lea        TextAlt,a0          ; Adresse vom alten Text
 lea        TextNeu,a1          ; Adresse des neuen Textes
Schleife:                       ; Anfang der Schleife
 move.b     (a0)+,d1            ; Wert holen
 cmp.b      #"a",d1             ; Vergleich mit "a"
 blo        Normal              ; kleiner ?
 cmp.b      #"z",d1             ; Vergleich mit "z"
 bhi        Normal              ; größer
 sub.b      #("a"-"A"),d1       ; a-z umwandeln
Normal:                         ; Label für andere Zeichen
 move.b     d1,(a1)+            ; Zeichen speichern
 tst.b      d1                  ; nicht unbedingt notwendig
 bne        Schleife            ; Nein, dann weiter
 rts                            ; Ja, dann Ende

TextAlt:                        ; alter Text
 dc.b       "Umwandlung in Großschrift. "
 dc.b       "Sonderzeichen wie $ * # _ @ ... "
 dc.b       "werden nicht veraendert."
 dc.b       0                   ; Ende

TextNeu:                        ; Speicher für Umwandlung
 ds.b       200

 end
```

Der 68000er kann auf Worte und Langworte nur zugreifen, wenn diese an geraden Adressen liegen. Dies gilt auch für Befehle. Legt man keine Daten in Byte-Form im Programm ab, kann auch nichts schiefgehen. Muß man aber Texte oder andere Dinge verwenden, die auf Bytes angewiesen sind, kann es passieren, daß andere Daten, die hinter den Bytes liegen, an einer ungeraden Adresse liegen. Sollte dies der Fall sein, brauchen Sie auf den Guru meistens nicht lange zu warten. Aber was ist gegen diesen aufdringlichen Blinker zu tun? Kein Problem für einen Assembler. Wenn Sie hinter Byte-Daten den Befehl

```
even
```

setzen, kann gar nichts schiefgehen. Even fügt ein zusätzliches Byte ein, wenn die Adresse ungerade ist. Ist die Adresse durch Zufall schon gerade, ändert Even nichts am Programm-Code. Sie sollten sich angewöhnen, Even hinter allen Byte-Daten zu verwenden!

```
DC.W 11,270,34,23,4535,2435
DC.B "Test",0
EVEN
DS.W 100
DS.B 77
DS.B 100
EVEN
```

Nun haben Sie schon ein ganz beachtliches Wissen über Assembler. Im BASIC-Teil des Buches haben wir ja schon gezeigt, was mit den Routinen alles zu machen ist. Nebenbei werden wir auch noch weitere Befehle kennenlernen. Sollte Ihr Interesse, in Assembler auch weiterhin zu programmieren, geweckt sein, sollten Sie unbedingt ein Buch über den Motorola 68000 und diverse Bücher zum Betriebssystem des Amiga kaufen. Für alle, die sich gut in Englisch verständigen können, sind die offiziellen Entwicklungshandbücher von Commodore sicher gut geeignet. Sie sind jedoch nicht ganz billig. Möchten Sie gerne deutsche Bücher zu diesem Thema kaufen, kann ich Ihnen die beiden Intern-Bände von DATA BECKER empfehlen. Auch sie sind mit sachlichen Informationen gefüllt - und dazu noch erheblich billiger. Umfangreiche Information mit vielen praxisnahen Beispielen in Assembler finden Sie auch im Maschinensprache-Buch.

In BASIC haben wir die nötigen Routinen über die .bmap-Files zugänglich gemacht. Diese Dateien enthielten die nötigen Hilfen, um die verschiedenen

Funktionen mit Parametern über ihren Namen aufrufen zu können. In Assembler können wir auch mit Dateien arbeiten, die den .bmap-Files ähnlich sind. Das eigentliche Programm lädt diese zwar nicht nach, dafür kann sie aber der Assembler verwenden. Auf der Profimat-Diskette befinden sich die ASCII-Files, in denen die Betriebssystemeinsprünge abgelegt sind, im Includes-Verzeichnis. Sie tragen den Namen der Library, für die sie zuständig sind. Weiterhin sind sie mit der Endung ".Offsets" versehen.

Bevor man auf eine Library zugreifen kann, muß man sie öffnen. Das ist in BASIC so gewesen - und es wird in Assembler nicht anders sein. Mit einer Ausnahme: Die exec.library ist immer verfügbar. Sie ist über eine festgelegte Adresse zu erreichen. Aber warum ist ausgerechnet diese Bibliothek immer verfügbar? Die Antwort ist logisch: weil man Exec braucht, um andere Libraries zu öffnen. Sie enthält die elementarsten Funktionen. In Verbindung mit Exec kommt man mit einer der wenigen Adressen in Berührung, die beim Amiga immer gleich ist. Es ist die Adresse 4. In dieser Adresse ist ein Zeiger auf die exec.library abgelegt.

Man definiert daher in jedem Programm zuerst diese Adresse:

```
ExecBase=4
```

Nun kann man den Zeiger komfortabel auslesen:

```
MOVE.L ExecBase,a6
```

könnte man z.B. schreiben, um die Adresse von Exec in das Register A6 zu befördern. Warum ausgerechnet A6? Nun, das hat Commodore so festgelegt. Bevor man eine Funktion aus einer Bibliothek aufruft, muß man deren Basisadresse immer in A6 ablegen. Zum einen deshalb, um überhaupt auf die normale Art einspringen zu können, und zum anderen, weil viele Funktionen die Library-Adresse in A6 erwarten. Der eigentliche Aufruf erfolgt dann über einen Unterprogrammaufruf. Unterprogramm? Das haben wir in Assembler doch noch gar nicht gemacht. Richtig, aber wir holen es jetzt nach.

Unterprogramme werden, wie eingangs erwähnt, mit RTS verlassen. Um ein Unterprogramm aufrufen zu können, muß man JSR verwenden. JSR ist mit dem BASIC-Befehl GOSUB zu vergleichen.

Prinzipiell läuft das so ab:

```
Hauptprogramm:
 ...
 JSR Unterprogramm
 ...
 JSR Unterprogramm
 ...
 RTS

Unterprogramm:
 ...
 RTS
```

6.1.5 Zugriff auf die Bibliotheken

Die Funktionen des Betriebssystems werden nicht über ihre Adresse, sondern über einen Offset (= Distanzwert) aufgerufen. Dieser Offset ist in den ".Offsets"-Dateien zu finden. Er ist immer negativ. Das stört uns aber nicht. Uns interessiert viel mehr, wie wir die Routinen aufrufen. Dazu gibt es ein einfaches Schema:

```
MOVE.L LibBase,A6
JSR FunktionsOffset(A6)
```

LibBase steht stellvertretend für die Speicherstelle, in der die Adresse der Bibliothek abgelegt ist (z.B. ExecBase). Mit FunktionsOffset legen wir fest, welche Funktion wir aufrufen möchten. Hat man vorher Zuordnungen zwischen Funktionsname und Offset vorgenommen oder die ".Offsets"-Dateien eingebunden, kann man an dieser Stelle einfach den Namen der Funktion angeben. In den Files der Profimat-Diskette ist allen Namen "_LVO" vorgestellt. Ein Aufruf sähe dann z.B. so aus:

```
MOVE.L LibBase,A6
JSR _LVOFunktionsName(A6)
```

Wir werden im weiteren Verlauf dieses Kapitels die Offsets für die Funktionen immer im Programm definieren. Damit auch die Leser die Programme verwenden können, die den Profimat nicht besitzen. Wir werden _LVO nicht verwenden.

Zuerst werden wir über die Exec-Library versuchen, eine weitere Bibliothek zugänglich zu machen. Haben wir das geschafft, liegen uns keine Steine mehr im Weg.

Die Funktion, die wir benötigen, trägt den Namen OpenLibrary(). Ihr Offset ist -552.

Damit auch die richtige Bibliothek geöffnet wird, müssen wir noch den Namen und die Versionsnummer übergeben. Verwenden wir die Versionsnummer Null, wird die Library, wenn verfügbar, immer geöffnet. Ihre Version wird dann nicht überprüft.

Ganz wichtig ist es, alles, was man in einem Programm öffnet, auch wieder zu schließen. Spätestens, wenn man mit Dateien arbeitet, sollte man diesen Vorsatz beherzigen, da man sonst so manche unangenehme Überraschung erleben kann. Nach Gebrauch wollen wir aber auch die Libraries schließen. Die Funktion dazu nennt sich CloseLibrary() und hat den Offset -414.

Als Parameter muß man die Adresse der Library übergeben, die man schließen möchte. Man erhält diesen Wert von OpenLibrary.

Parameter sind Werte, die eine Funktion braucht, um ihre Aufgabe erfüllen zu können. Rufen Sie z.B. eine Funktion auf, die eine Datei von Diskette löscht, müssen Sie dieser Funktion als Parameter einen Zeiger auf den Namen übergeben, den Sie löschen wollen. In BASIC haben wir die Parameter einfach in einer Klammer hinter dem Funktionsnamen übergeben. Auch erhielten wir manchmal einen Rückgabewert, den wir mit "=" in einer Variable sicherten. In Assembler müssen wir etwas anders vorgehen. Alle Parameter müssen in Registern abgelegt werden. Welcher Parameter in welches Register gehört, kann man der Entwickler-Dokumentation, Amiga Intern und anderen Büchern entnehmen. Für die Funktionen, die wir im weiteren Verlauf dieses Buches aufrufen, werden wir Ihnen selbstverständlich sagen, wie Sie die Parameter zu übergeben haben. Sicherheitshalber sollten Sie immer Langworte verwenden. Die Entwickler-Dokumentation schweigt sich nämlich über die gewünschte Datenlänge aus. Doch nun zu einem möglichen Ergebnis. Liefert eine Routine einen Rückgabewert, liegt dieser immer in D0.

Setzen wir das neu Gelernte doch gleich mal in ein Programm um:

```
; ************************************************
; *                                              *
; *  Programm: OpenLibraryTest                   *
; *                                              *
; *  Mirko Langlotz                              *
; *                                              *
; *  01.09.1988                                  *
; *                                              *
; ************************************************

ExecBase = 4          ; Basis von exec.library

OpenLibrary = -552    ; Funktionen aus exec.library
CloseLibrary = -414
```

Um zu testen, ob auch wirklich etwas passiert, rufen wir am besten eine Funktion aus einer neu geöffneten Bibliothek auf. Es ist ratsam, eine sehr einfache Funktion zu nehmen, um den Überblick nicht zu verlieren. Eine einfache und immer aufrufbare Funktion ist DisplayBeep() aus Intuition. Sie sorgt dafür, daß einer oder alle Screens kurz aufblitzen. Meist setzt man diese Funktion ein, um einen Fehler zu signalisieren. Wir benutzen sie, um einen Beweis dafür zu haben, daß wir tatsächlich auf Intuition zugegriffen haben. Auf alle Fälle müssen wir, da wir die ".Offsets"-Files nicht verwenden, den Offset von DisplayBeep festlegen.

```
DisplayBeep = -96     ; Funktion aus intuition.library
```

Nun beginnen wir das eigentliche Programm. Nach dem anfänglichen Label legen wir gleich richtig los. Um eine neue Bibliothek zu verwenden, müssen wir OpenLibrary aus Exec anspringen. Dazu holen wir die Adresse von Exec nach A6. Bevor wir jedoch in die Funktion einspringen können, müssen wir noch festlegen, was eigentlich geöffnet werden soll. Wir übergeben daher einen Zeiger auf den Namen der Library in A1. Der Name muß unbedingt klein geschrieben werden! Sonst stürzt OpenLibrary() ab. Für die von der Funktion verlangte Funktionsnummer setzen wir Null ein, womit wir signalisieren, daß wir mit allem zufrieden sind, was verfügbar ist. Das heißt - eigentlich machen wir es aus Faulheit...

```
Programm:
 move.l     ExecBase,a6        ; Zeiger auf Exec
 lea        IntName,a1         ; Name von Intuition
 move.l     #0,d0              ; Version
```

Nun ist der große Moment gekommen. Wir setzen JSR ein, um eine Exec-Routine auszuführen. Vergessen Sie nicht (a6) einzugeben! Eine Guru-Meditation würde Ihnen sonst die Vergeßlichkeit heimzahlen.

```
jsr         OpenLibrary(a6)     ; Öffnen (a6=ExecBase)
```

OpenLibrary() liefert als Ergebnis einen Zeiger auf die Library. Dieser Zeiger kann später genauso verwendet werden, wie ExecBase. Sollte die Bibliothek nicht aufzufinden sein, was bei Intuition eigentlich nie der Fall sein dürfte, erhält man NULL zurück. Im diesem Fall haben Sie sich wahrscheinlich bei der Eingabe des Namens vertippt oder das abschließende Null-Byte vergessen, das das Ende des Strings markiert. Um keinen Absturz zu riskieren, überprüfen wir dies mit TST und BEQ.

```
tst.l       d0                  ; Rückgabe testen
beq         Fehler              ; Fehler (=0) ?
```

Wenn alles geklappt hat, sichern wir D0. Sie müssen auf den abgelegten Wert immer dann zurückgreifen, wenn Sie eine Funktion aus der zugehörigen Library ausführen wollen.

```
move.l      d0,IntBase          ; OK, dann sichern
```

Nun rufen wir ganz demonstrativ eine Funktion aus Intuition auf. DisplayBeep haben wir uns auserkoren. Sie verlangt in A0 einen Zeiger auf den Screen, der aufblitzen soll. Übergibt man statt dessen NULL, blitzen alle vorhandenen Screens. (Meist ist das ohnehin nur die Workbench.) Da wir auch keine Screen-Adresse kennen, verwenden wir NULL. Wichtig ist auch, den in IntBase gesicherten Zeiger auf Intuition nach A6 zu laden. Dann können wir endlich unsere erste Routine aus Intuition in Assembler aufrufen.

```
move.l      IntBase,a6          ; Zeiger auf Intuition
move.l      #0,a0               ; Parameter
jsr         DisplayBeep(a6)     ; Aufruf (a6=IntBase)
```

Nach getaner Arbeit wollen wir unsere Library - ordentlich wie wir nun mal sind - wieder schließen. CloseLibrary() aus Exec macht's möglich. Die Adresse der Library (nicht die des Namens) muß in A1 zu finden sein. Vergessen Sie nicht, A6 mit dem richtigen Wert (ExecBase) zu füllen.

```
move.l      ExecBase,a6    ; Wieder auf exec zugreifen
move.l      IntBase,a1          ; Parameter: IntBase
jsr         CloseLibrary(a6)    ; intuition.library
                                ; schließen
                                ; (a6=ExecBase)
```

Mit dem Programm sind wir fertig. Nur noch ein kleines RTS, und wir haben alles, was nötig ist, getan. Oder doch nicht? Die Daten fehlen nämlich noch.

```
Fehler:
 rts                            ; Programm verlassen
                                ; wird auch bei Fehler
                                ; angesprungen
```

Hierzu ist nicht mehr viel zu sagen. Bei IntName legen wir die Bezeichnung von Intuition ab. Zur Markierung des Textendes packen wir noch ein Null-Byte dran. Mit EVEN richten wir die aktuelle Adresse. Dann brauchen wir nur noch Platz für einen Zeiger zu reservieren und ein End anzufügen. So einfach ist das.

```
IntName:    dc.b "intuition.library",0
                           ; Wichtig : klein schreiben
 even                      ; Adresse begradigen

IntBase:    ds.l 1         ; Ein Langwort für Zeiger
                           ; auf Intuition

 end
```

In diesem Programm - es ist übrigens das erste, das ein sichtbares Ergebnis (wenn auch nur eine kurze Farbveränderung) liefert, und daher nicht mit dem Debugger verfolgt werden braucht - haben wir einmal

```
MOVE.L #0,D0
```

und

```
MOVE.L #0,A0
```

verwendet. Man kann den ersten dieser Befehle durch

```
CLR.L D0
```

ersetzen. Haben Sie schon erraten, welche Funktion CLR hat? Richtig, der Operant (in diesem Fall D0) wird gelöscht, also mit Null initialisiert. Leider kann man CLR nicht auf Adreßregister anwenden. Doch auch Adreßregister kann man elegant auf Null setzen. Für A0 sähe eine Lösung so aus:

```
SUB.L A0,A0
```

Wir erzählen Ihnen das nicht etwa, weil der Amiga so langsam ist oder weil wir am Speicherplatz sparen möchten. Vielmehr wollen wir Ihnen helfen, Programme zu verstehen, die in anderen Büchern und Zeitschriften abgedruckt sind. Dort werden diese Befehle nämlich gerne verwendet. Bei so erstaunlichen Konstruktionen wie SUB.L A0,A0 brauchen Sie also nicht ins Schwitzen zu kommen. Denken Sie sich einfach, dort stünde MOVE.L #0,A0.

Ein ganz wichtiger Befehl ist JMP. JMP ist ein Sprungbefehl, der im Gegensatz zu den uns bereits bekannten Bcc-Befehlen immer verzweigt. In BASIC würde man ihn GOTO nennen.

JSR und JMP können unter bestimmten Umständen durch BSR und BRA ersetzt werden. Da diese Befehle im Vergleich zu JSR und JMP nur wenige Adressierungsarten zulassen, werden wir sie nicht verwenden.

Besitzer früherer Rechner konnten sich freuen, wenn ihr Gerät einigermaßen schnell addieren konnte. Für den Amiga ist das ein Kinderspiel.

```
ADD.W VAR1,VAR2     ; VAR2=VAR2+VAR1

SUB.W VAR1,VAR2     ; VAR2=VAR2-VAR1
```

Richtig auf Touren kommt er erst, wenn er multipliziert und dividiert. Dazu braucht man kein Programm und keine Routine - der 68000er kann das ganz allein. Er kann auf Wunsch sogar mit Vorzeichnen rechnen. Teilt er, stellt er nicht nur das eigentliche Ergebnis, sondern auch den Rest zur Verfügung. Dadurch kann man leicht eine Routine schreiben, die mit Nachkommastellen arbeiten kann. Nachdem wir uns die theoretische Grundlage dazu erarbeitet haben, werden wir ein Programm erstellen, das die ersten Primzahlen berechnet. Doch zunächst die Theorie. Beginnen wir mit der Multiplikation. Egal, ob man mit oder ohne Vorzeichen rechnet, verwendet man immer das gleiche Format. Der Befehl für normale Multiplikation lautet MULU. Für vorzeichenbehaftete Zahlen verwendet man MULS. Die beiden Operanden werden als Worte interpretiert. Das Ergebnis hat Langwortgröße. Beispiel gefällig?

```
MULS D0,D1

D1.L = D1.W * D0.W
```

Nach der Multiplikation folgt dann die Division. Dabei wird das Langwort des Zieloperanden durch das Wort des Quelloperanden geteilt. Der Quelloperand ist die erste, der Zieloperand die zweite Angabe hinter dem Befehl. Das bei der Rechnung entstehende Ergebnis wird als Wort im Zielregister abgelegt. Als Zielregister kann man nur ein Datenregister verwenden. Ein Blick in die Tabellen des Profimat beweist das. Der bei der Division entstehende Rest wird in der oberen Hälfte des Zielregisters gelegt. Bei den Divisionsbefehlen darf man nur eines nicht tun: durch Null teilen. Warum das so ist, wollen wir einmal beiseite lassen. Dividiert man durch Null, ruft der 68000er eine Exception-Routine auf. Diese Routine produziert, wenn man Sie nicht ändert, leider einen Guru.

Will man den Rest, den die Divison liefert, verwenden, muß man diesen Rest in die untere Hälfte des Registers bringen. Dazu kann man verschiedene Verfahren anwenden. Am einfachsten ist jedoch der SWAP-Befehl. Er tauscht die obere und untere Hälfte des Datenregisters aus, das als Operand angegeben wird.

Noch ein Befehl ist in Verbindung mit den Divisionen interessant. Es geht um EXT. Dieser Befehl erweitert ein Register unter Berücksichtigung des Vorzeichens auf doppelte Breite.

```
EXT.W macht aus einem Byte ein Wort.
EXT.L verwandelt ein Wort in ein Langwort.
```

Rechnet man mit sehr großen Zahlen ohne Vorzeichen, sollte man EXT nicht verwenden. Man muß dann die obersten Bits löschen, oder ein Datenbyte oder -wort in ein gelöschtes Wort oder Langwort kopieren. Wie man einzelne Bits oder Bitgruppen bearbeitet, nehmen wir später noch unter die Lupe.

So, jetzt wissen wir alles, um unser Primzahlprogramm zu schreiben. Dabei gibt es ein kleines Problem. Wie sollen wir die errechneten Zahlen ausgeben? Wir lösen es, indem wir die berechneten Zahlen in ein DOS-Fenster ausgeben. Diese Methode ist die einfachste. Bei dieser Gelegenheit müssen wir noch eine Kleinigkeit klären.

6.1.6 Die richtige Eröffnung

Wie Sie wissen, kann man ein Programm sowohl über das CLI als auch über die Workbench starten. Für Anwender gibt es da auch keine Zweifel. Doch es gibt einige Programme, die nur laufen, wenn sie vom CLI gestartet werden. Die CLI-Befehle können z.B. nicht von der Workbench gestartet werden. Auch dann nicht, wenn man ihnen ein Icon verpaßt. Das liegt an mehreren Dingen. Am Augenscheinlichsten ist jedoch die Tatsache, daß ein von der Workbench gestartetes Programm kein Fenster bekommt, um darin etwas auszugeben. Neben diesem oberflächlichen Problem muß sich ein Programmierer noch mit ganz anderen Dingen herumschlagen.

Auch wenn man keine direkte Parameterübergabe an ein Programm vorsieht, muß man sich um verschiedene Dinge kümmern. Erfolgte der Programmaufruf von der Workbench-Ebene, sendet die Benutzeroberfläche eine Message (= Meldung) an das gestartete Programm. Diese Meldung muß das Programm abwarten, bevor es mit seiner eigentlichen Aufgabe beginnen kann. Bevor man es über RTS beendet, muß man die Workbench über diesen Schritt informieren.

In unseren bisherigen Programmen haben wir uns um diese Dinge nicht gekümmert. Funktioniert hat aber trotzdem alles. Damit ist es nun vorbei. Bestimmte Funktionen des Betriebssystems erwarten nämlich, daß man sich darum gekümmert hat, die Meldung abzuwarten. Sollte man es unterlassen haben, stürzen diese Routinen ohne erkennbaren Grund ab. Warum sie das tun, ist schwer zu verstehen. Die Absturzkandidaten sind bevorzugt in der Library zu finden, die für Dateien und Geräte zuständig ist: DOS. Da wir unsere Werte über ein DOS-Fenster ausgeben wollen, müssen wir die Workbench-Meldung auswerten.

Sie brauchen übrigens keine Angst zu haben, jetzt bei jedem Programm darüber zu grübeln, wie man die ankommende Meldung übernehmen muß. Wir haben für Sie vorgearbeitet und ein Programmgerüst erarbeitet, mit dem nichts mehr schief gehen kann. Wenn Sie ein eigenes Programm schreiben möchten, laden Sie dieses Gerüst einfach ein und fügen Ihre Routinen an. Wenn Sie wollen, können Sie unser Programm noch dahingehend erweitern, daß automatisch alle wichtigen Bibliotheken geöffnet werden. So sparen Sie sich viel Arbeit. Hier nun unser Programmgerüst:

```
; ************************************************
; *                                              *
; *  Programm:                                   *
; *                                              *
; *  Sprache:                                    *
; *                                              *
; *  Autor:                                      *
; *                                              *
; *  Datum:                                      *
; *                                              *
; ************************************************

ExecBase = 4

FindTask = -294       ; Task suchen
WaitPort = -384       ; Auf Meldung warten
GetMsg   = -372       ; Meldung holen
ReplyMsg = -378       ; Meldung zurück schicken
Forbid   = -132       ; Andere Tasks stoppen

; Hier können weitere Offsets eingegeben werden

SysStart:
 move.l     ExecBase,a6              ; Adresse von eigenem
 move.l     #0,A1                    ; Task suchen
 jsr        FindTask(a6)
 move.l     d0,Task

 move.l     d0,a0                    ; Wurde Programm
 tst.l      172(a0)                  ; von Workbench oder
 bne        VonCLI                   ; CLI gestartet ?

VonWorkbench:
 move.l     a0,-(sp)
 lea        92(a0),a0
 move.l     ExecBase,a6
 jsr        WaitPort(a6)
 move.l     (sp)+,a0
 lea        92(a0),a0
 move.l     ExecBase,a6
 jsr        GetMsg(a6)
 move.l     d0,WB_Msg
VonCLI:

 jsr        Programm                 ; Eigentliches Programm
                                     ; ausführen

 tst.l      WB_Msg                   ; Wenn Programm von
```

```
 beq        ProgrammEnde            ; Workbench gestartet
                                    ; wurde, Meldung
                                    ; zurück schicken

 move.l     ExecBase,a6
 jsr        Forbid(a6)
 move.l     WB_Msg,A1
 move.l     ExecBase,a6
 jsr        ReplyMsg(a6)

ProgrammEnde:
 clr.l      d0
 rts

WB_Msg:     DC.L 0
Task:       DC.L 0

Programm:                   ; Ab hier kann das eigentliche
 rts                        ; Programm abgelegt werden!

 end                        ; Nicht vergessen
```

Seien Sie bitte nicht böse, wenn wir es unterlassen, das obenstehende Programm zu erklären. Warum es funktioniert ist für Sie an dieser Stelle unwichtig - wichtig ist nur, daß es funktioniert. Wenn Sie dennoch näheres wissen wollen, können Sie dies in Amiga Intern Band II nachlesen.

6.1.7 Tempo ist angesagt: Primzahlen

Wenden wir uns wieder der Primzahlberechnung zu. Wir werden ein relativ einfaches System verwenden. Zuerst formulieren wir das Programm in BASIC. Das könnte z.B. so aussehen:

```
"****************************************
"**                                    **
"**  Berechnung von Primzahlen in BASIC  **
"**                                    **
"****************************************

PRINT 2 "Die erste Primzahl ist 2
FOR Zahl=3 TO 49999& STEP 2
   FOR Teiler=2 TO SQR(Zahl)
      IF Zahl/Teiler=INT(Zahl/Teiler) THEN GOTO NZahl
```

```
   NEXT Teiler
   PRINT Zahl
NZahl:
NEXT Zahl
END
```

Alle Primzahlen bis 49999 werden ausgegeben. Die erste Primzahl (zwei) geben wir von Hand aus. Das ergibt einen Geschwindigkeitsvorteil, da wir immer gleich in Doppelschritten weitergehen können. Es kann nämlich keine gerade Primzahl größer zwei geben. Dann sehen wir nach, ob wir einen Teiler finden, durch den man unsere Zahl ohne Rest teilen kann. Werden wir fündig, kann es sich nicht um eine Primzahl handeln. Wir testen die Teiler jedoch nur bis zur Wurzel der gesuchten Zahl durch. Dadurch erzielen wir einen erheblichen Geschwindigkeitszuwachs. Wenn wir in diesem Bereich keinen Teiler finden, kann es auch oberhalb der Wurzel keinen geben, da der Wert, der bei einer Division oberhalb der Quadratwurzelgrenze als Ergebnis auftauchen würde, unterhalb diese Grenze schon einmal als zulässiger Teiler in Erscheinung getreten sein müßte. Wenngleich sich diese Aussage etwas verwirrend anhört, stimmt sie doch. Das müssen Sie uns schon glauben.

Überlegen wir uns nun, wie wir obenstehendes Programm in Assembler umsetzen. Probleme stehen uns außer der Berechnung der Wurzel nicht im Haus. Nicht vergessen dürfen wir die Ausgabe. Dazu müssen wir, wie bereits erwähnt, die Meldung abfangen, die von der Benutzeroberfläche geschickt wird.

```
; ************************************************
; *                                              *
; *  Programm: Primzahl.ASM                      *
; *                                              *
; *  Sprache:  Assembler / Profimat              *
; *                                              *
; *  Autor:    Mirko Langlotz                    *
; *                                              *
; *  Datum:    01.09.1988                        *
; *                                              *
; ************************************************

ExecBase = 4

FindTask = -294       ; Task suchen
WaitPort = -384       ; Auf Meldung warten
GetMsg   = -372       ; Meldung holen
ReplyMsg = -378       ; Meldung zurück schicken
Forbid   = -132       ; Andere Tasks stoppen
```

```
OpenLibrary = -552                    ; Exec
CloseLibrary = -414

Open = -30                            ; Dos
Close= -36
Write = -48

SysStart:
 move.l     ExecBase,a6               ; Adresse von eigenem
 move.l     #0,A1                     ; Task suchen
 jsr        FindTask(a6)
 move.l     d0,Task

 move.l     d0,a0                     ; Wurde Programm
 tst.l      172(a0)                   ; von Workbench oder
 bne        VonCLI                    ; CLI gestartet ?

VonWorkbench:
 move.l     a0,-(sp)
 lea        92(a0),a0
 move.l     ExecBase,a6
 jsr        WaitPort(a6)
 move.l     (sp)+,a0
 lea        92(a0),a0
 move.l     ExecBase,a6
 jsr        GetMsg(a6)
 move.l     d0,WB_Msg
VonCLI:

 jsr        Programm                  ; Eigentliches Programm
                                      ; ausführen

 tst.l      WB_Msg                    ; Wenn Programm von
 beq        ProgrammEnde              ; Workbench gestartet
                                      ; wurde, Meldung
                                      ; zurück schicken

 move.l     ExecBase,a6
 jsr        Forbid(a6)
 move.l     WB_Msg,A1
 move.l     ExecBase,a6
 jsr        ReplyMsg(a6)

ProgrammEnde:
 clr.l      d0
 rts
```

```
WB_Msg:     DC.L 0
Task:       DC.L 0
```

Das war die Pflicht - jetzt kommt die Kür. Zuerst kümmern wir uns um ein Fenster. Dazu öffnen wir die DOS-Bibliothek.

```
Programm:
 lea        DosName,a1
 clr.l      d0
 move.l     ExecBase,a6
 jsr        OpenLibrary(a6)
 move.l     d0,DosBase
 beq        Fehler_0
```

Dann kommt das Fenster. Ein Fenster wird von DOS wie eine Datei über einen Dateinamen und die Open-Funktion geöffnet. In D1 wird die Adresse des Namens und in D2 ein Zugriffsmodus übergeben. 1005 ist für Lesen vorgesehen. Mit 1006 kann man schreiben. Da wir jedoch ein Fenster und keine Datei auf einem Datenträger öffnen, können wir 1005 verwenden, obwohl wir eigenlich Daten ausgeben. Über Fenster kann man sowohl Daten ausgeben als auch Texte einlesen. Open() übergibt in D0 einen Zeiger, über den die Datei dann weiterhin angesprochen wird. Im Falle eines Fehlers wird wieder Null zurückgegeben.

```
 move.l     #DateiName,d1
 move.l     #1005,d2
 move.l     DosBase,a6
 jsr        Open(a6)
 move.l     d0,File
 beq        Fehler_1
```

In BASIC haben wir die Geschwindigkeit durch SQR gesteigert. Auch in Assembler wollen wir auf diesen Luxus nicht verzichten. Wir werden die Wurzel aber nicht direkt, sondern anhand einer Tabelle, die die Quadrate der Zahlen 0-999 angibt, ermitteln. Diese Tabelle müssen wir zunächst einmal anlegen.

```
 clr.w      d0
 lea        Quadrat,a0
 move.l     a0,a1
Wurzel:
 move.w     d0,d1
 mulu       d1,d1
 move.l     d1,(a0)+
 add.w      #1,d0
```

```
 cmp.w      #500,d0
 bne        Wurzel
```

Nun kommen wir zur eigentlichen Hauptschleife. Wir verwenden hier unter anderem auch die neuen Befehle BTST und BSET. Wir werden sie beschreiben, nachdem wir mit dem Programm fertig sind.

```
 move.l     #2,d0

Test:
 btst       #0,d0
 bne        KannSein
 cmp.l      #2,d0
 beq        ZahlenAusgabe
 bset       #0,d0
KannSein:
```

Wir ermitteln die Wurzel, indem wir die Tabelle Schritt für Schritt durchgehen. Dieses Verfahren ist zwar nicht sehr elegant - uns reicht es jedoch. Professioneller wäre es z.B., in der Mitte der Liste zu beginnen und dann beide Hälften weiter zu teilen, bis man mit wenigen Vergleichen den richtigen Wert gefunden hat.

```
 clr.w      d1
 clr.w      d4
Suchen:
 add.w      #1,d1
 add.w      #4,d4
 cmp.l      (a1,d4),d0
 bhi        Suchen
```

Dann testen wir alle Teiler durch. Finden wir einen solchen, kann es sich nicht um eine Primzahl handeln. In diesem Fall suchen wir weiter.

```
 move.w     #2,d2
WeiterTeilen:
 move.l     d0,d3
 divu       d2,d3
 swap       d3
 tst.w      d3
 beq        NZahl
 add.w      #1,d2
 cmp.w      d1,d2
 bls        WeiterTeilen
```

Gab es aber keinen Teiler, geben wir die Zahl mit einer noch zu schreibenden Routine aus. Bevor wir diese aufrufen, retten wir aber noch die Register, deren Inahlt wir noch benötigen. Wir könnten sie einzeln auf den Stack legen. Zur Abwechslung verwenden wir jedoch MOVEM. Dieser Befehl kann mehrere Register auf dem Stack ablegen und wieder von ihm runterholen.

```
ZahlenAusgabe:
 movem.l    d0/a0-a1,-(sp)
 jsr        Ausgeben
 move.l     d0,d1
 movem.l    (sp)+,d0/a0-a1
 tst.l      d1
 beq        Fehler_2
```

Bei 50000 soll unsere Schleife enden. Wir schließen noch das Fenster und die dos.library, dann verlassen wir das Programm.

```
NZahl:
 add.l      #1,d0
 cmp.l      #50000,d0
 blo        Test

Fehler_2:
 move.l     File,d1
 move.l     DosBase,a6
 jsr        Close(a6)

Fehler_1:
 move.l     DosBase,a1
 move.l     ExecBase,a6
 jsr        CloseLibrary(a6)

Fehler_0:
 rts
```

Die Routine zur Ausgabe der Zahl fehlt uns noch:

```
Ausgeben:
; Die meiste Zeit ist der Amiga damit beschäftigt, die
; Werteauszugeben. Fügen Sie an dieser Stelle einen RTS-
; Befehl ein, können Sie die reine Rechenzeit ermitteln.
; Wenn Sie weiterhin bedenken, daß man durch ein anderes
; Berechnungssystem die Geschwindigkeit noch weiter
; steigern kann, sehen Sie, wie schnell der Amiga ist,
; wenn er in Assembler direkt programmiert wird. BASIC
```

```
; oder Compiler dürften da wohl nicht mehr mithalten
; können. Vergleichen Sie doch einmal die Rechenzeit des
; BASIC-Programms (ohne PRINT) mit der Geschwindigkeit
; dieses Programms (mit RTS) !!!!
 lea        Hilfe+7,a0
 move.b     #$a,-(a0)
 move.w     #10,d1
NStelle:
 move.l     d0,d2
 divu       d1,d2
 swap       d2
 move.b     d2,d4
 add.b      #"0",d4
 move.b     d4,-(a0)
 swap       d2
 ext.l      d2
 move.l     d2,d0
 bne        NStelle
FillSchleife:
 cmp.l      #Hilfe,a0
 beq        OkToWrite
 move.b     #" ",-(a0)
 jmp        FillSchleife
OkToWrite:
 clr.w      d0
 move.l     File,d1
 move.l     #Hilfe,d2
 move.l     #7,d3
 move.l     DosBase,a6
 jsr        Write(a6)
 rts
```

Schließlich müssen wir noch einen Text ablegen und Speicher reservieren. Dann haben wir es endlich geschafft.

```
DateiName:  dc.b     "RAW:0/0/200/200/Primzahlen",0
DosName:    dc.b     "dos.library",0
Hilfe:      ds.b     10
 even
DosBase:    ds.l     1
File:       ds.l     1
Quadrat:    ds.l     500

 end
```

Wenn Sie unser Primzahlenprogramm noch nicht verstehen, ist das nicht weiter schlimm. Es ist auch nicht ganz einfach und soll auch nur dem Zweck

dienen, Ihnen die Möglichkeiten der Sprache aufzuzeigen. Wenn Sie richtig in Assembler einsteigen wollen, werden Sie in den meisten Fällen ohnehin nicht umherkommen, sich spezielle Bücher zu diesem Thema zu kaufen. Vielleicht ziehen Sie es aber auch vor, lieber über AmigaBASIC auf das System zuzugreifen. Auch C könnte eine Alternative sein. Näheres zu C erfahren Sie in Kapitel 6.2.

Wir wollen Ihnen, bevor wir das Assembler-Kapitel beenden, noch ein paar Befehle an die Hand geben, die Ihnen wichtige Aufgaben abnehmen. Zuerst wären da die Bit-Befehle. Mit Ihnen kann man ein einzelnes Bit testen oder ändern.

Um festzulegen, welches Bit des Zieloperanden man ändern will, kann man beim Quelloperand dessen Nummer entweder unmittelbar oder über ein Datenregister angeben.

Befehl	Funktion
BTST	Angegebenes Bit ins Z-Flag kopieren
BCLR	Bit löschen
BSET	Bit setzen
BCHG	Bit invertieren. (0->1; 1->0)

Angenehm ist auch der MOVEM-Befehl. Mit ihm kann man mehrere Register der gleichen Datenlänge auf den Stack legen. MOVEM holt diese Werte später auch wieder zurück. Die gewünschten Register kann man mit einem Schrägstrich voneinander trennen. Liegen sie in einer Reihe, kann man auch einen Bereich mit "-" angeben.

```
D0/D1/D2/D3 entspricht D0-D3
```

Hier nun ein Beispielaufruf von MOVEM:

```
...
MOVEM.L D0-D3/D5/A0/A2,-(SP)
...
MOVEM.L (SP)+,D0-D3/D5/A0/A2
...
```

Schließlich, wollen wir Sie noch in ein kleines Geheimnis der Amiga-Programme einweihen. Wie Sie wissen, können Sie dreimal für jedes Programm

einstellen, wohin es geladen werden soll. Diese drei Teile wurden dort Code, Data und BSS genannt. Profimat kennt nun auch drei gleichnamige Befehle. Jeder von ihnen sorgt dafür, daß der nachfolgende Quellcode in den Bereich assembliert wird, der durch den Befehl angegeben ist. Die Programmausführung beginnt im Code-Segment (Code-Teil). BSS ist eine Besonderheit. Was Sie in dieses Segment assemblieren, wird nicht abgespeichert. Es wird lediglich im Programm eine Information darüber abgelegt, wie lang dieser Teil ist. Beim Laden wird dann der dafür nötige Speicher reserviert.

Dieses Verfahren ist sehr vorteilhaft. Braucht eines Ihrer Programme z.B. 100000 Bytes Speicher, können Sie diesen einfach mit DS mit BSS-Segment belegen. Das Programm wird dadurch auf der Diskette nicht länger!

6.2 C

Die Programmiersprache C erfreut sich auf dem Amiga immer größerer Beliebtheit. Dies mag zum einen daran liegen, daß sie sehr stark an Pascal angelehnt ist, und ist zum anderen daran zu sehen, daß größere Teile des Amiga-Betriebssystems in C programmiert wurden.

Wir möchten mit diesem Kapitel über C einen Einblick in die Programmierung dieser Sprache bieten. Das zweite Anliegen ist es, dem mit der Sprache vertrauten Programmierer einige Kniffe zu zeigen, die ihm helfen, besser mit C zu arbeiten.

Da C eine Compiler-Sprache ist, erfordert dies einiges Umdenken. Ein C-Programm wird mit einem einfachen Editor geschrieben, der nur einen ASCII-Text bearbeitet, der mit einem Brief zu vergleichen ist. Es gibt zwar besondere Editoren, die hinsichtlich der Übersichtlichkeit einige Vorteile bieten, aber schon der CLI-Editor Ed reicht aus, um in C einigermaßen ordentlich zu programmieren.

Ist ein Programm-Text erstellt, wird der Compiler gestartet. Dieses Programm übersetzt in unserem Fall den Programm-Text in einen zweiten Programm-Text. Dieser unterscheidet sich vom Vorgänger dadurch, daß er nicht in der Sprache C, sondern in Assembler verfaßt ist. Wie Sie aus dem Assembler-Teil vor diesem Kapitel wissen, muß auch dieser Text über einen Assembler übersetzt werden, und zwar liegt jetzt ein als Objekt-Code bezeichnetes File vor.

Dieses ist der Programm-Text in der für den Computer verständlichen Form. Allerdings muß man diesen beim Amiga noch über einen Linker in eine ladbare Form bringen, die dann das endgültige Produkt unserer Arbeit darstellt.

Sie sehen also, daß man drei Schritte zu überwinden hat, wenn man vom Programm-Text zu lauffähigen Programmen kommen möchte. Dies alles kann aber nur mit einem gewissen Komfort abgewickelt werden, wenn die Arbeitsumgebung richtig geschaffen ist. Der Weg führt uns deshalb unweigerlich zum Einrichten der Arbeits-Diskette.

6.2.1 Installation des Systems

Wenn Sie den C-Compiler als Programm-Paket erhalten - wir wollen hier vom Aztec-C-Compiler reden - finden Sie mindestens zwei Disketten vor. Auf der einen befinden sich alle für die Arbeit wichtigen Dinge, und auf der anderen sind einige Beispiele untergebracht.

Zum Erstellen einer Arbeits-Diskette fertigen Sie zuerst eine Kopie der Compiler-Diskette an. Dies ist die erste Diskette, die sich auch durch Booten starten läßt. An dieser Kopie wollen wir dann im Verlauf einige Veränderungen vornehmen und besprechen, wozu eigentlich dieses oder jenes gebraucht wird.

Damit wir von gleichen Voraussetzungen reden, ist hier eine Auflistung des Directory-Inhalts meiner Kopie:

```
lib (dir)
Trashcan (dir)
c (dir)
l (dir)
devs (dir)
s (dir)
t (dir)
fonts (dir)
libs (dir)
include (dir)
Empty (dir)
bin (dir)
Expansion (dir)
```

Wie Sie sehen, enthält die Diskette alle notwendigen Verzeichnisse, um als Boot-Diskette bzw. Workbench-Diskette genutzt zu werden. Dies ist auch beabsichtigt, denn gerade beim Programmieren in C benötigt man ein vollständig ausgestattetes System.

Beim genauen Betrachten des Directories fallen einige bisher unbekannte auf. Gehen wir sie, eins nach dem anderen, durch: Über Trashcan, Empty und Expansion muß wohl kein Wort mehr verloren werden. Auch t und fonts sind für uns in der Besprechung uninteressant.

Wichtiger sind schon die Standard-Verzeichnisse wie z.B. C, in dem ja alle CLI-Befehle vorhanden sind. Da wir später unter akuter Platznot leiden werden, ist es schon einmal nicht unwichtig, wenn man hier nachsieht, welche der CLI-Befehle wirklich gebraucht werden. Manche sind unbedingt notwendig, um auch nur die einfachsten Arbeiten zu erledigen, andere werden je nach Konfiguration überhaupt nicht gebraucht. Diese sollten Sie ruhig löschen, denn schließlich kann man sie immer noch wieder von der normalen Workbench-Diskette zurückholen. Hier ist der gekürzte Inhalt meines C:-Verzeichnisses von der Arbeits-Diskette:

```
AddBuffers      Assign
BindDrivers     Break
CD              ChangeTaskPri
Copy            Date
Delete          Dir
DiskChange      e
Echo            Ed
Editor          Else
EndCLI          EndIf
FailAt          Fault
If              Info
Install         Join
Lab             List
LoadWb          make
MakeDir         Mount
NewCLI          Path
Quit            Rename
Run             set
SetClock        SetMap
Skip            Sort
Stack           Status
Type            Version
Wait
```

Ich war bei der Verteilung an dieser Stelle noch recht großzügig, denn nur wenige CLI-Kommandos mußten weichen. Hinzugekommen sind allerdings zwei Kommandos, die gar nicht zum Standard gehören, sondern vom Compiler stammen. Es sind Make und Set, beides Befehle, die zur Einrichtung des Compiler-Systems und zur Arbeit damit gebraucht werden. Wir gehen später noch ausführlicher darauf ein.

Das S:-Verzeichnis muß vorhanden sein, weil in ihm beim Booten die Start-up-Sequence gesucht wird. Etwas anderes ist dort auch nicht vorhanden!

Gleichermaßen wichtig für das System im allgemeinen sind l und libs. Beide Verzeichnisse enthalten Programm-Segmente, die im Falle eines Aufrufes nachgeladen und deshalb dringend benötigt werden. Würde man diese Verzeichnisse entfernen, liefe das System zwar immer noch, jedoch gäbe es an manchen Stellen einfach Rückschläge, weil ein benötigter Teil, der auf der Diskette zu suchen war, nicht gefunden wird. So lassen sich dann später Libraries nicht öffnen oder Disketten nicht validatieren.

Überaus üppig ausgestattet ist das DEVS:-Verzeichnis. Es benötigt ebensowenig die Keymaps wie die Druckertreiber. Alles dies kann getrost bis auf die eine deutsche Keymap und den richtigen Druckertreiber entfernt werden, damit wir wieder Platz schaffen. Hier ist wieder einmal der Zustand meines Verzeichnisses aufgelistet:

```
    keymaps (dir)
      d
    printers (dir)
      Epson
    clipboards (dir)
clipboard.device                MountList
narrator.device                 parallel.device
printer.device                  serial.device
system-configuration
```

An dieser Stelle sei noch darauf hingewiesen, daß genausowenig, wie man irgendwelche Libraries löscht, die von Diskette nachgeladen werden, natürlich auch kein Device gelöscht werden sollte, denn ohne Printer.Device ist es nicht einmal möglich, ein Listing auf dem Drucker auszugeben! Mehr läßt sich also hier nicht machen. Nun kommen wir zu den drei übrig gebliebenen Verzeichnissen, die ausschließlich dem Compiler vorbehalten sind. "lib" enthält Programmteile, die beim schon erwähnten Linken des Programms noch hinzuge-

fügt werden. Hier sind Funktions-Aufrufe und mathematische Berechnungen untergebracht, die man so getrost vernachlässigen kann. Die libs sind allerdings auf jeden Fall auf der Diskette zu lassen, weil ohne sie kein Programm zum Laufen gebracht werden kann.

Wenn man auch nur eine Zeile mit dem Amiga-Betriebssystem programmiert, kommt man um die Include-Files nicht herum. Sie beinhalten vorgefertigte Definitionen und Strukturen, die bei der Programmierung der Libraries, Devices und aller anderen System-Komponenten unentbehrlich sind. Also läßt sich auch hier nichts kürzen.

Das letzte Directory mit dem Namen "bin" enthält die Compiler-Packet-Programme. Neben dem Compiler-, Assembler- und Linker-Programm ist hier der Set-Befehl zu finden, den ich im CLI-Verzeichnis untergebracht habe. Weiterhin sind hier jeweils ein Programm zum Setzen des Datums und eins zur Anzeige des freien Speichers. Beide sind recht hilfreich, wenn man keine anderen hat. Entscheiden Sie selbst, was Ihnen besser gefällt.

Nach dieser Zusammenstellung der wichtigsten Verzeichnisse und deren Daten kommen wir jetzt zu den neuralgischen Punkten, z.B. zur Startup-Sequence! Sie zeichnet gerade ein gut eingerichtetes Compiler-Paket aus. Nur wer hier seine Einstellungen richtig organisiert, hat später bei der Arbeit wenig Probleme. Als Beispiel sehen Sie sich hier meine Startup-Sequence an, bei der wir jede Einstellung und jedes Vorgehen besprechen werden. Je nach Voraussetzungen (Anzahl der Diskettenlaufwerke, Speicher, Festplatte, resetfeste RAM-Disk) sind bestimmte Veränderungen vorzunehmen.

```
; Startup-Sequence der Aztec-3.6a-Arbeitsdiskette
; für allgemeine Anwendungen
; ========================================================
Stack 8000                          ; Arbeitsspeicher
AddBuffers SYS: 25                  ; Beschleunigung der
AddBuffers DF1: 25                  ; Diskettenoperationen
AddBuffers DF2: 25
Echo "*NA2000  Workbench 1.3*N"
Echo "Aztec-C 3.60a Workdisk Wgb*N"
SetClock opt load                   ; Systemuhr neu setzen
SetMap d
MakeDir RAM:lib                     ; Arbeitsverzeichnisse
                                    ; erstellen
MakeDir RAM:Aztec
MakeDir RAM:t
MakeDir RAM:s
```

```
Copy SYS:lib/#?.lib to RAM:lib   QUIET ; LIBs kopieren
Copy SYS:Aztec/#?   to RAM:Aztec QUIET
                                   ; Aztec-Files kopieren
Copy SYS:(m|d)      to RAM:      QUIET ; Batchfiles in
                                       ; die RAM-Disk
Copy SYS:s/.editorinit to RAM:s
Copy copy     to RAM:c
Copy C:Editor to RAM:Ed
                                ; Editor mit mehr Komfort
Path RAM:Aztec
                                ; Aztec-Files als Kommandos
SET CLIB=RAM:lib/ INCLUDE=SYS:include CCTEMP=RAM:
Assign g: RAM:              ; Aktuelles Verzeichnis im RAM
Assign s: RAM:s
Cd RAM:
E df1:Start-Sequence        ; Spezielle Einrichtungen für
                            ; die Arbeitsdiskette
```

Wie Sie gleich am Anfang meiner Startup-Sequence sehen, besitze ich drei Laufwerke. Dies ist für die Arbeit mit Compilern empfehlenswert. Aber gehen wir Schritt für Schritt die Kommandos durch:

```
Stack 8000
```

Der Stack ist der Speicher, der beim Start eines CLI-Befehls oder Programms reserviert wird, um Daten zwischenzuspeichern. Dies ist je nach Befehl oder Programm unterschiedlich. Die Befehle benötigen höchsten 1024 Bytes. Ein Programm kann, wenn es sehr verschachtelt programmiert ist, noch einiges mehr fordern. Deshalb habe ich zur Sicherheit den Stack erhöht. Sie sehen aber spätestens, wenn ein Programm, in dem kein Fehler steckt, abschmiert, daß der Stack zu klein bemessen war.

```
AddBuffers SYS: 25
```

Dieses und die folgenden Kommandos erhöhen den Pufferspeicher der Diskettenlaufwerke. Das macht den Zugriff in geeignetem Maße schneller. Allerdings ist hier Vorsicht geboten, denn jedes Laufwerk verbraucht 20 KByte Speicher. 25 weitere Puffer-Blöcke erhöhen den Verbrauch auf 32 KByte. Die Frage, die Sie sich stellen sollten, ist, ob der Speicherverbrauch in Ihrem System überhaupt möglich ist. Nur wenn Sie mehr als 512 KByte haben, ist das empfehlenswert!

Die Textausgabe, die auf die drei Kommandos folgt, kann sicherlich entfallen. Man sollte aber nicht ohne Grund darauf verzichten, denn diese Nachricht teilt einem zu Anfang mit, um welche Diskette es sich handelt. Schließlich besitzt man ja mehrere bootfähige Disketten, und da kann eine Verwechslung schon mal vorkommen.

```
SetClock opt Load
```

Für diejenigen, die eine interne akku-gepufferte Uhr besitzen, ist dieser Befehl gedacht. Er lädt in die Amiga-Uhr den Zeitwert. Besitzen Sie keine akku-gepufferte Uhr, können Sie den Befehl entfernen. Es empfiehlt sich dann aber, den normalen Befehl

```
DATE ?
```

einzugeben, um das Datum zu stellen. Der Aztec-C-Compiler nutzt die Uhrzeit und das Datum mit seinen Hilfsprogrammen. Es ist gerade deshalb ratsam, bei der Programmierung immer mit den richtigen Zeitwerten zu arbeiten.

Besitzen Sie eine andere Uhr, müssen Sie natürlich den dafür vorgesehenen Befehl eingeben.

Als Nächstes wird die deutsche Tastaturbelegung geladen. Achten Sie darauf, daß die Tastatur, die Sie auswählen, alle Sonderzeichen enthält, die bei C-Programmen gebraucht werden. Auf keinen Fall dürfen die runden, eckigen und geschweiften Klammern fehlen (), [], {}. Aber auch das eingestrichene Anführungszeichen ist für Characters besonders wichtig ("). Es befindet sich auf dem kleinen "ä" mit der Tastenkombination Alt-ä.

```
MakeDir
```

Darauf folgt ein größerer Block, in dem neue Verzeichnisse im RAM eingerichtet werden. Diese Verzeichnisse werden am Anfang beim Booten des Systems in das RAM mit ihrem kompletten Inhalt geladen. Dadurch beschleunigt sich der spätere Zugriff. Das ist allerdings nur bei einem Speicher von mindestens 1 Mega-Byte möglich, weil sonst der Arbeitsspeicher zu knapp wird. Eine andere Alternative dazu ist die Anschaffung einer schnellen Festplatte, bei der der Zugriff ähnlich schnell wie aus dem RAM stattfindet. Ist beides nicht vorhanden, bleibt nur der Weg über die Diskettenstation.

Als Verzeichnisse werden RAM:lib und RAM:Aztec eingerichtet. In Lib sind die Aztec-Libraries für Mathematik und die Standard-Befehle zu finden. Ich verwende nur die einfach-genauen Versionen. Das Aztec-Verzeichnis wird für die drei Befehle verwendet: den Compiler, Assembler und Linker. Sicherlich könnte man die drei genau wie die Libraries auf der Diskette lassen, aber weil diese gerade bei kleinen Programmen immer wieder geladen werden müssen, habe ich sie ins RAM geholt.

Die Verzeichnisse T und S werden von meinem Editor benötigt. Ich verwende nicht Ed von der Workbench-Diskette, sondern den Editor aus dem "Großen C-Buch". Er beherrscht einige vorzügliche Funktionen wie das Herausfalten von Textpassagen oder das Strukturieren des Textes. Außerdem ist er programmierbar, was ihn für aufwendige Veränderungen besonders beliebt macht. Wie gesagt, er benutzt diese beiden Verzeichnisse.

```
Copy
```

Nach dem Einrichten der Verzeichnisse müssen dann auch die Files kopiert werden. Das erledigen die beiden Kopier-Befehle.

Die nächsten Kopier-Befehle holen Batch-Dateien ins RAM, bzw. die init-Datei des Editors und den Editor selbst, der im RAM als Ed liegt, damit ich keinen neuen Befehl lernen muß (man ist ja so schwer umzugewöhnen).

6.2.2 Die ersten Schritte in C

Um die ersten Schritte in C machen zu können, müssen wir uns erst das Konzept dieser Programmiersprache vorhalten. Es heißt "Kompatibilität". Also soll ein Programm, das wir auf dem Amiga in C schreiben auch auf anderen Computern, die die Sprache C erlauben, lauffähig sein.

Dies wird dadurch erreicht, daß man den Programmtext nicht in einer vorcodierten Form niederschreibt, sondern ihn als einfaches Text-File auf der Diskette abspeichert. Es ist quasi so, als schriebe man einen Brief. Diesen Text übersetzt dann unser Compiler in einen Assembler-Quell-Code, der für den jeweiligen Prozessor verständlich ist. Dieser wiederum wird vom Assembler (sie lasen darüber im vorhergehenden Kapitel) in einen Programm-Code

übertragen. Mit diesem Code-Modul müssen wir über den Linker nur noch ein lauffähiges Programm erstellen. Fertig!

Dieser Weg, den wir hier skizziert haben, ist nötig, um einen beliebigen Quelltext auf beliebigen Computern zum Laufen zu bringen. Aber machen wir uns keine Probleme, die wir nicht haben. Unser Ziel ist das Programmieren in C auf dem Amiga 500.

Also beginnen wir mit unserem ersten Programm. Das sieht wie folgt aus:

```
main()
   {
   printf("Das große Amiga 500 Buch\n");
   }
```

Listing 6.2.1: Das erste Programm

Programmbeschreibung

Dieses Programm muß zuerst mit

```
ed Test.c
```

über den CLI-Editor eingegeben werden. Dann verlassen Sie den Editor mit [Esc]-[X] und das Programm wird auf der Diskette gespeichert. Nun rufen Sie den Compiler auf:

```
cc Test
```

Die Endung .c muß nicht angegeben werden, da der Compiler dies als selbstverständlich annimmt. Treten keine Fehler auf - dazu kommen wir später noch - ruft der Compiler automatisch den Assembler auf, den man eigentlich sonst mit

```
as Test
```

aufrufen müßte. Aber so sparen Sie Arbeit. Zuletzt muß dieses Assembler-Modul noch als Programm gelinkt werden. Dafür verwenden wir den Aztec-Linker, der folgendermaßen gestartet wird:

```
ln Test -lc
```

Neben dem Programmnamen müssen wir noch die Option -l ergänzen, die eine Linker-Library einbindet. In der C-Library sind alle erweiterten C-Kommandos untergebracht. Da printf() schon ein erweiterter Befehl ist, müssen wir seine Definition dem Linker mitteilen (sonst wäre dieser nicht in der Lage, unser Programm zu erstellen). Jetzt müßte aber alles fehlerfrei abgelaufen sein.

Sie können nun mit der einfachen Eingabe von

```
Test
```

unser Beispiel-Programm aufrufen, das den oben stehenden Text ausgibt. Aber was ist nun eigentlich mit dem Listing? Was bedeutet "main", was die Klammern dahinter, welche besondere Bedeutung haben die geschweiften Klammern und warum muß der Text zu den Anführungszeichen auch noch in Klammern gesetzt werden?

Viele Fragen haben wir - und fast alle haben etwas mit den Klammern zu tun. Daran läßt sich leicht zeigen, daß in der Sprache C die Klammern, ob rund, eckig oder geschweift, eine große Rolle spielen.

Die Sprache C ist eine modulare Sprache. Sie verwendet zur Programmgestaltung ein Hauptmodul. Neben diesem können noch weitere kleinere Untermodule existieren, die wie die Unterprogramme in BASIC verstanden werden können. Das Hauptmodul trägt den Namen "main". Damit ist der erste unbekannte Begriff aus dem Beispiel-Programm geklärt.

Die Befehle jedes Moduls stehen in geschweiften Klammern und werden so als zusammengehörig gekennzeichnet. Deshalb sind in den Zeilen vor und hinter dem printf() die beiden Klammern.

Auch mit den beiden Klammern (diesmal sind es runde) hinter dem Funktions-Namen main hat es eine besondere Bewandtnis. In ihnen werden normalerweise Parameter angegeben, die dem Modul main übergeben werden können. Da aber nichts übergeben werden soll, sind die Klammern leer.

Anders sieht das beim Befehl printf() aus. Wir haben es in diesem Fall nämlich genauso mit einem Modul zu tun wie auch bei main. Diesem Modul, das

in C als Funktion bezeichnet wird, übergeben wir einen Wert, unseren Beispiel-Text. Allerdings müssen wir die Funktion nicht erst programmieren, sie wird in der Bibliothek des Compilers mitgeliefert.

Halten wir zum Abschluß noch einmal fest:

- ein C-Programm zeichnet sich durch seine Module aus, die man als Funktionen bezeichnet
- die Main-Funktion entspricht dem Hauptprogramm in anderen Programmiersprachen
- alle Befehle einer Funktion (bzw. eines Blocks) werden in geschweifte Klammern gesetzt, um sie deutlich von anderen Befehlen abzugrenzen
- es gibt auch Funktionen, die man nicht selbst programmiert, die man aber trotzdem benutzen kann. Sie werden mit dem Compiler mitgeliefert
- Parameter werden einer Funktion in runden Klammern übergeben, dies zeigt sich sowohl beim Funktionsaufruf wie auch bei der Definition

Lassen Sie uns mit dem erhaltenen Wissen gleich etwas programmieren. Wir werden dazu analog zum ersten Beispiel-Programm ein neues schreiben, bei dem eine neue Funktion definiert wird:

```
main()
   {
   ...
   }
...
```

zuerst brauchen wir wieder das Hauptprogramm (Entschuldigung), die main()-Funktion. Außerdem definieren wir uns zuvor noch eine eigene Funktion mit dem Namen Ausgabe():

```
Ausgabe()
   {
   printf("Die Funktion Ausgabe arbeitet!\n");
   }
```

Im Hauptprogramm können wir die selbst definierte Funktion jetzt aufrufen. Dort schreiben wir

```
{
printf("Hauptprogramm\n");
Ausgabe();
}
```

Das gesamte Programm sieht jetzt so aus (für diejenigen, die es so zerstückelt, wie es abgedruckt war, nicht abtippen möchten):

```
Ausgabe()
   {
   printf("Die Funktion Ausgabe arbeitet!\n");
   }
main()
   {
   printf("Hauptprogramm\n");
   Ausgabe();
   }
```

Listing 6.2.2: Eine selbst-definierte Funktion

Programmbeschreibung

Obwohl die Funktion Ausgabe() vor dem Hauptprogramm definiert ist, wird sie nicht wie bei BASIC-Programmen ausgeführt. Der Compiler erkennt selbständig, daß es sich um ein "Unterprogramm" handelt und startet nur das Hauptprogramm.

In diesem wird zuerst ein Text über die Compiler-Funktion printf() ausgegeben. Dann kommt der große Moment! Wir rufen Ausgabe() auf. Die Ausführung setzt sich in der Funktion fort. Auch dort steht ein printf()-Kommando, das natürlich auch ausgeführt wird. Weil kein Befehl mehr kommt, springt die Funktion zurück zum Hauptprogramm, und auch dort findet sich kein Kommando mehr, so daß die Ausführung beendet ist.

Nebenbei: Bei dieser Gelegenheit sollte Ihnen eine Besonderheit der Programmiersprache C aufgefallen sein. Am Ende der Befehlszeilen findet sich immer wieder ein Semikolon. Dieses wird hier genau wie bei Pascal zur Tren-

nung zweier Kommandos benutzt. Es steht also immer am Ende eines Befehls. Niemals darf es in der Mitte eines Befehls zu finden sein. Man darf es aber auch nicht vergessen, weil sonst der Compiler nicht weiß, wo ein Befehl aufhört.

Sehen wir uns jetzt an, welche Variablen-Typen C besitzt, damit wir auch Berechnungen durchführen können. C unterscheidet zuerst zwischen ganzzahligen Werten und Fließkommazahlen. Beide gibt es in zwei Varianten: einfacher Wertebereich und doppelter Wertebereich.

Ganzzahlige Werte bezeichnet man als Integer. Der Variablen-Typ heißt deshalb

```
int
```

Bei doppelt großem Wertbereich haben wir es mit "langen" Zahlen zu tun. Daher leitet sich die Definition ab:

```
long int
```

oder gleich

```
long
```

Die Fließkommazahlen besitzen auch zwei Typenbezeichnungen: Die erste heißt

```
float
```

und steht für den einfachen Typ. Die zweite setzt sich wieder aus dem vorangestellen LONG zusammen und kann aber auch durch einen weiteren Begriff ersetzt werden:

```
long float
```

oder dafür

```
double
```

Es gibt neben diesen Zahlen-Typen natürlich noch andere Variablen. Darauf kommen wir auch noch, aber jetzt sollte erst einmal ein Programm die Theo-

rie zur Praxis werden lassen. Unser Programm definiert drei Variablen. Zweien weist es Werte zu und der dritten das Ergebnis einer Berechnung:

```
main()
   {
   long float a, b, c;
   a = 3.141596;
   b = 1.48;
   c = b * a;
   printf("Das Ergebnis: %f\n", c);
   }
```

Listing 6.2.3: Berechnungen in C

Programmbeschreibung

Wie Sie sehen, unterscheidet sich das Rechnen in C nicht besonders vom Rechnen in anderen Programmiersprachen. Etwas anders als z.B. in BASIC ist allerdings die Variablen-Deklaration. Zu Anfang der Funktion Hauptprogramm müssen nämlich die Variablen, die man verwenden möchte, definiert werden. A, b und c sind also vom Typ double. Alle Berechnungen werden mit doppelter Genauigkeit ausgeführt.

Am Schluß findet sich dann eine Ausgabe, die wir zwar schon gesehen haben. Allerdings tritt hier eine Besonderheit auf. Die Variable c, die im Text ausgegeben werden soll, wird nicht in ihn eingebaut. Dafür steht dort ein Platzhalter ("%f") und nach dem Text die Variable. Die Funktion printf() baut bei der Ausgabe des Textes den Wert an dieser Stelle ein. Probieren Sie es aus!

Dieses Programm muß, weil es Fließkommazahlen verwendet, auch mit der Mathematik-Library des Compilers gelinkt werden. Der Befehl lautet deshalb ln Berechnungen -lm -lc!

Neben den numerischen Variablen besitzt die Sprache C aber noch eine Variable, mit der sich Texte speichern lassen. Es handelt sich um den Datentyp

```
char
```

der ein Zeichen des ASCII-Zeichensatzes in sich aufnehmen kann. Das bringt uns allerdings nicht so viel, wenn wir ganze Worte oder Sätze speichern wol-

len. Dafür nimmt man einfach ein Feld aus diesen Einzelzeichen. Das folgende Programm-Beispiel wird Ihnen das verdeutlichen.

```
char Test[] = "Hallo, ich bin der Amiga!\0";
main()
   {

   printf("%s\n", &Test[0]);

   }
```

Listing 6.2.4: Text als Variablen

Programmbeschreibung

Am Anfang dieses neuen Listings wird ein Zeichen-Feld definiert. Es hat den Namen Text und besitzt eine vom Compiler zu berechnende Länge. Im Hauptprogramm steht uns diese Variable als globale Variable zur Verfügung, d.h. sie ist für alle Funktionen - nicht nur die main()-Funktion - definiert. Das Hauptprogramm macht nun nichts anderes, als daß es diesen Text über printf() ausgibt. Dafür verwenden wir einen neuen Platzhalter. "%s" steht für einen String, dessen Anfangsadresse, ermittelt durch &Text[0], nach dem eigentlichen Text der Funktion übergeben wird. Diese fügt dann den Text in den String der Ausgabe ein.

Wie Sie sehen, ist es zwar einfach, mit festen Texten zu arbeiten, aber die Probleme kommen erst, wenn man mit Texten variabler Länge umgehen muß. Dieses Problem ist natürlich von den C-Programmierern und den Entwicklern der Compiler-Pakete schon längt erkannt worden. Deshalb finden wir auch in der Bibliothek, die wir immer mit -lc dazulinken, einige Funktionen, die das Arbeiten mit Strings erleichtern.

So kopiert die Funktion strcpy() einen String in einen anderen. Das sieht dann so aus:

```
strcpy(Ziel, Quelle);
```

wobei Ziel und Quelle POINTER auf Strings sind. Darunter versteht man die Startadressen der Texte, wie wir sie mit dem "&"-Zeichen bekommen haben. Definiert werden die Variablen folgendermaßen:

```
char *Ziel, *Quelle;
```

Der Stern vor den beiden Variablen-Namen kennzeichnet, daß es sich nicht um die Variablen selbst handelt, sondern um die Zeiger (Speicheradresse) auf den Inhalt. Das bringt uns natürlich ein Problem: Wir haben zwei Zeiger auf Texte, die wir aber noch nicht haben. Wir brauchen noch Felder, in denen die wirklichen Texte zu finden sind:

```
char Text1[20], Text2[20];
```

Nun kann z.B. ein Programm zur Eingabe eines Textes auffordern, der im ersten String-Feld abgelegt wird. Dieser wird in das zweite Feld kopiert und dann ausgegeben. Hier ist das Listing:

```
main()
   {
   char Text1[20], Text2[20];
   printf("Den Text bitte: ");
   scanf("%s", &Text1[0]);
   strcpy(&Text2[0], &Text1[0]);
   printf("%s war die Eingabe!\n", &Text2[0]);
   }
```

Listing 6.2.5: String-unterstützende Funktionen

Programmbeschreibung

Zuerst definieren wir zwei Textfelder, dann wird der Benutzer - also Sie - zur Tastatur gebeten. Die Funktion scanf() ähnelt dem von BASIC her bekannten INPUT-Befehl. Hier wird aber wie bei printf() zuerst ein Text übergeben, der Informationen oder Aufforderungen enthalten kann. Außerdem müssen in ihm auch noch die entsprechenden Platzhalter für die einzulesenden Variablen eingefügt werden. Das Ergebnis dieser Aktion ist, daß in dem Feld Text1 die Eingabe untergebracht ist. Beachten Sie, daß nur 20 Zeichen erlaubt sind!

Nun tritt die Funktion in Kraft, deretwegen wir das Programm geschrieben haben. Strcpy() macht seinem Namen hoffentlich alle Ehre und transferiert den Inhalt des ersten Feldes in das zweite Feld. Um dies zu überprüfen, wird am Ende noch der Inhalt des zweiten Feldes ausgegeben. Das war die erste String-Funktion!

Eine andere, aber genauso wichtige String-Funktion ist das Vergleichen von Strings, das man entweder selbst programmieren kann oder aber auch vom Compiler-Paket übernimmt. Es empfiehlt sich die Compiler-Version, weil sie einerseits Standard und zweitens bestimmt auch schneller ist. Der Name der Funktion ist

```
strcmp(Str1, Str2);
```

Sie übergeben wie bei der ersten Funktion die Adressen zweier Strings. Dies geschieht entweder über zwei Pointer auf Zeichenketten oder die Adresse des ersten Feldelements. Wichtig ist aber im Gegensatz zur strcpy()-Funktion, daß wir hier ein Ergebnis erwarten, das nicht in einer der beiden Variablen abgelegt wird. Hier gibt die Funktion selbst das Ergebnis zurück. Die Definition wird folgendermaßen erweitert:

```
int strcmp(Str1, Str2);
char *Str1, *Str2;
```

Damit legen wir im Programm fest, daß die Funktion strcmp() zwei Pointer auf Zeichenketten übergeben bekommt und dafür einen Integer-Wert zurückliefert. Probieren wir dies gleich in einem Programm aus. Dafür nehmen wir ganz einfach zwei fest definierte Strings, so sparen wir sogar noch die Eingabe. Diese werden verglichen und das Ergebnis, das strcmp() zurückliefert, geben wir mittels printf() aus.

```
char String1[] = "Amigo\0";
char String2[] = "Amiga\0";
main()
   {
   int Ergebnis;
   Ergebnis = strcmp(String1, String2);
   printf("Ergebnis: %d\n", Ergebnis);

   }
```

Listing 6.2.6: Zwei Strings vergleichen

Programmbeschreibung

Unser Programm definiert zu Anfang zwei globale Strings, die sich in ihrem letzten Buchstaben unterscheiden. In der Hauptroutine wird dann über die Variable Ergebnis der Wert der neuen Funktion strcmp() abgefragt. Das Ergebnis bekommen wir dann auf dem Bildschirm ausgegeben.

Zum Test sollten Sie ruhig einmal die Namen verändern oder gleichsetzen, so daß ein anderes Ergebnis entsteht. Wir wollen dann nämlich als nächstes über Abfragen im Programm eine Fallunterscheidung vornehmen.

Zum Abfragen von Bedingungen stellt uns C genau wie das AmigaBASIC die Konstruktion

```
IF ... THEN ... ELSE
```

zur Verfügung. Allerdings kennt die Sprache C kein THEN, was aber nicht so viel macht. Die Konstruktion wird hier nämlich mit Befehlsblöcken organisiert. Das sieht dann in der Theorie so aus:

```
if (Bedingung)
   {
   im Falle des Falles
   }
else
   {
   sollte der Fall nicht eingetreten sein
   }
```

Als schlaue Programmierer setzen wir dies schnell in den entsprechenden Programmtext um. Dieser soll zwei eingegebene Strings auswerten und je nach Gleichheit eine Meldung ausgeben:

```
main()
   {
   char Text1[20], Text2[20];
   int Ergebnis;
   printf("Den 1. Text bitte: ");
   scanf("%s", &Text1[0]);
   printf("Den 2. Text bitte: ");
   scanf("%s", &Text2[0]);
   Ergebnis = strcmp(Text1, Text2);
   if (Ergebnis == 0L)
```

```
    {
    printf("Die Strings sind gleich!\n");
    }
else
    {
    printf("Leider gibt es kleine Unterschiede!\n");
    }
}
```

Listing 6.2.7: Zwei Strings vergleichen und auswerten

Programmbeschreibung

Damit haben wir uns auch in das Gebiet der unterscheidenden Programme vorgewagt. Eigentlich ist es doch gar nicht so schwer. Zumindest, wenn man sich etwas mit den AmigaBASIC-Kommandos auskennt.

Zu bedenken ist, daß alle Befehle, die innerhalb der geschweiften Klammern stehen, ausgeführt werden, wenn die Bedienung erfüllt ist bzw. nicht erfüllt werden konnte (else).

Wir können hier also wieder einen neuen Befehlsblock bilden, der mehrere Kommandos nacheinander enthält.

Hier zeigt sich auch, warum das Einrücken bei jedem Befehlsblock so wichtig ist. Denn treten größere Verschachtelungen auf, unterstützen die Tabulatoren die Lesbarkeit erheblich.

Programmstrukturen

Als Abschluß der Einführung in die Programmiersprache C möchte ich Sie mit den Programmstrukturen vertraut machen. Neben der IF-THEN-ELSE-Konstruktion gibt es noch weitere Elemente, die das Programmieren in C so einfach machen. Sehen wir uns deshalb als erstes die Schleifen an.

Wie in BASIC kennt auch C eine FOR-NEXT-Schleife. Sie ist zwar am Anfang etwas anders aufgebaut, arbeitet aber genau gleich.

Schematisch sieht das Schleifengebilde etwa so aus:

```
for(Anfang; Ende; Schritte)
   {
   /* Befehle */
   }
```

Die for()-Funktion, die am Anfang des Befehls-Blocks steht, gibt an, welchen Start- und welchen End-Wert die Laufvariable haben muß und welche Schritte dazwischen ausgeführt werden. Versuchen wir zuerst, eine aus dem Amiga-BASIC bekannte Schleife umzusetzen:

```
FOR i = 1 to 20

   "hier stehen sonst ein paar Befehle
NEXT i
```

Diese Schleife sieht - übersetzt in C - dann so aus:

```
for(i=1; i<=20; i++)
   {
   /* die Befehle */

   }
```

Innerhalb der Argument-Klammer sehen wir drei Angaben. Die erste stellt den Start-Wert ein. Hier können Sie der Laufvariablen ihren Anfangs-Wert zuweisen. Es lassen sich aber durchaus mehrere Operationen durch Kommata trennen, wenn es nötig ist, z.B. zwei Variablen am Anfang zu initialisieren.

Als zweites haben wir die Bedingung, die erfüllt sein muß, damit die Schleife weiterhin durchlaufen wird. In diesem Beispiel muß erst ein Maximum von 20 überschritten werden, bis der Schleifendurchlauf beendet ist.

Wie aber kommt die Variable auf diesen Wert? Bei BASIC-Befehlen ist eine Schrittweite von 1 eingestellt, die bei jedem Durchlauf zur Laufvariablen addiert wird. In C gibt es diesen Standard nicht; er muß vom Benutzer gewählt werden. Aus diesem Grund steht als letzter Befehl in den Klammern die Anweisung, die bei jedem neuen Durchlauf ausgeführt werden soll: in diesem Fall das Erhöhen der Variablen i um jeweils einen Schritt.

Die FOR-Schleife ist also wie in BASIC zu handhaben. Sie hat zwar den Unterschied, daß schon am Kopf der Schleife alle Einstellungen gemacht werden müssen, aber dies hat mit dem Ergebnis nichts zu tun. Eine ganz andere Schleifenart, die uns aus dem AmigaBASIC leider nicht bekannt ist, stellt die WHILE-Schleife dar. Während eine FOR-Schleife mindestens einmal durchlaufen werden muß, denn erst am Ende findet der Test statt, ob weitere Durchläufe notwendig sind, kann es bei der WHILE-Schleife vorkommen, daß sie nicht ein einziges Mal abgearbeitet wird. Nur so lange, wie die Bedingung erfüllt ist, die man nach dem Befehl angibt, wird der Befehls-Bock ausgeführt.

```
while(Bedingung)
   {

   /* Befehle */
   }
```

Als Bedingung kann jede mathematische Relation angegeben werden. Ob man Variablen des Programms aus dem Speicher oder nackte Zahlen vergleicht, ist vollkommen unerheblich: solange der Vergleich mit dem Ergebnis als "wahr" bewertet werden kann, arbeiten die Befehle im danach folgenden Block. Ist die Bedingung unwahr, fährt das Programm danach fort.

Mit der gleichen Funktion kann auch eine genau umgekehrt angeordnete Schleife konstruiert werden. Das Kennzeichen der WHILE-Schleife war es ja, daß sie schon zu Anfang die Bedingung prüft und so manchmal nicht ein einziges Mal ausgeführt wird. Diese Überprüfung läßt sich aber auch an das Ende stellen, wenn man daraus eine DO-WHILE-Schleife macht:

```
do
   {

   /* Befehle */
   }
   while(Bedingung);
```

Das Schlüsselwort DO weist das Programm an, den nun folgenden Block auszuführen, bis ein Befehl zum Abbruch erteilt wird. Ob dieser Befehl gegeben werden muß, stellt der Test des WHILE-Befehls immer am Ende fest. Dieser Schleifentyp ist mit der REPEAT-UNTIL-Schleife aus Pascal verwandt, bei der das Wiederholkriterium auch erst am Ende getestet wird.

Es eignet sich besonders für Berechnungen, bei denen immer wieder nachgesehen werden muß, ob das Ziel schon erreicht wurde.

Endlosschleifen

Ein sehr interessantes Gebiet der Programmierung sind die Endlosschleifen. Sie eignen sich besonders für eine im Programm befindliche Hauptschleife.

So wird z.B. immer nach einem Befehl gefragt, der über die Tastatur eingegeben werden muß. Dann erfolgt ein Sprung in die Routine zur Auswertung, und am Ende fährt das Programm wieder mit der Eingabe fort. Das ist die Endlosschleife.

Solch eine Schleife läßt sich mit beiden Programmbefehlen in C programmieren. Versucht man es mit der FOR-Schleife, dann läßt man einfach die angeforderten Parameter weg und schreibt:

```
for(;;)
   {
   /* Befehle der Endlosschleife */
   }
```

Es handelt sich dann um eine sogenannte FORever-Schleife, die man auch unter dem definierten Begriff FOREVER aufrufen kann, wenn man ihn zuvor selbst einrichtet oder aber über ein Include-File eingerichtet hat:

```
#define FOREVER for(;;)
```

Der zweite Weg zu einer Endlos-Schleife bietet sich über die WHILE-Schleife an, bei der einfach nur eine Bedingung gefunden werden muß, die immer wahr ist. Das läßt sich ganz einfach machen, indem man den Wahrheitswert "wahr" als Argument angibt. Das sieht dann so aus:

```
while(TRUE)
   {
   /* Befehle der Endlosschleife */
   }
```

6.2.3 Größere und kleinere Beispielprogramme

Wir haben Sie im vorhergehenden Teil in groben Zügen mit den Grundlagen der C-Programmierung bekannt gemacht. Die wichtigsten Variablen-Typen, String-Operationen und bedingte Programmausführung waren die drei beherrschenden Themen des Kapitels.

Dieses Wissen benötigen Sie jetzt als Voraussetzung für unsere Beispielprogramme, die schon etwas anspruchsvoller als die vorhergehenden sein sollen. Zu den Themen gehört z.B. die Systemprogrammierung wie aber auch die geschickte C-Programmierung, bei der sich noch einige neue Türen für Sie öffnen werden.

Als erstes möchte ich Ihnen den Weg zu den System-Routinen ebnen. Von BASIC aus ließen sich die Routinen ganz einfach über den Befehl LIBRARY erreichen. Doch wir benötigten dazu ein .bmap-File. Das File ist in C nicht erforderlich, weil die Namen in dem c.lib-Modul gespeichert und auch die entsprechenden include-Dateien vorhanden sind.

Das Problem liegt hier vielmehr im Öffnen und Schließen der Libraries selbst. Während AmigaBASIC uns die gesamte Arbeit intern abnahm, haben wir hier die Hebel selbst in der Hand, aber wir müssen sie erst bedienen lernen.

Der Weg zu den System-Routinen führt über die System-Routinen. Welch ein Widerspruch! Aber es stimmt. Mit den System-Routinen der Exec.Library, die uns als C-Programmierer immer zur Hand stehen, können wir uns Zugang zu den Routinen aller anderen Libraries verschaffen.

Die äußerst wichtige Funktion heißt nämlich OpenLibrary() (ähnlich wie in BASIC der LIBRARY-Befehl) und gewährt uns den Zugriff (diesmal genau wie in BASIC) zu allen Funktionen der Librarys.

Hier ist ein Beispiel-Programm, das an das erste des BASIC-Kapitels angelehnt ist. Es öffnet die Intuition.Library und ruft die Funktion DisplayBeep() auf. Da hier keine genaueren Erklärungen bezüglich der Funktion selbst folgen, sollten Sie bei Unklarheiten noch einmal diesen Abschnitt im BASIC-Kapitel nachlesen.

```
/*************************************
 *                                   *
 * Programm: DisplayBeep() "Blinken" *
 * ================================= *
 *                                   *
 * Autor:  Datum:      Kommentar:    *
 * ------  ----------  ----------    *
 * Wgb     12.10.1988                *
 *                                   *
 *                                   *
 *************************************/
#include <intuition/intuition.h>
struct IntuitionBase *OpenLibrary(), *IntuitionBase;
main()
   {
   if (!(IntuitionBase =
                  OpenLibrary("intuition.library", 0L)))
      {
      printf("Keine Intuition-Library gefunden!\n");
      exit(FALSE);
      }
   DisplayBeep(0L);
   CloseLibrary(IntuitionBase);
   }
```

Listing 6.2.8: Library öffnen und Funktion aufrufen

Programmbeschreibung

Zu Anfang, bevor überhaupt eine Aktivität des Programms zu spüren ist, wird die Funktion OpenLibrary() definiert. Sie ist eine Exec-Funktion und daher jederzeit ansprechbar. Der Stern identifiziert diese Funktion als eine, die als Ergebnis einen Zeiger zurückliefert. Dieser Zeiger zeigt auf die Struktur mit dem Namen IntuitionBase. Leider haben wir diese gar nicht definiert, und man erwartet einen Fehler, der aber nicht kommt! Wir haben Vorsorge getroffen und mit #include eine sogenannte Kopf-Datei eingebunden, in der allerhand und unter anderem auch diese Struktur definiert ist. Für Sie ist wichtig, daß es sich um eine Strutur handelt, ohne die man die Funktionen der Intuition.Library nicht ansprechen kann. Es ist die Basis für die Arbeit mit Intuition (engl. base).

Nach dem Test, ob wir einen Zeiger auf die Basis erhalten haben, also die Funktionen angesprochen werden können, ruft das Programm die DisplayBeep()-Funktion auf. Der Bildschirm blinkt, und während wir das noch wahrnehmen, wird die Library schon wieder geschlossen. Dazu übergeben wir im Programm der CloseLibrary()-Funktion der Exec.Library den Zeiger auf die IntuitionBase.

Als nächstes möchte ich mit Ihnen ein Window öffnen. Schließlich gab es einen BASIC-Befehl, und es war über die Library-Funktion OpenWindow() möglich. In C ist das einerseits leichter als mit der Funktion in BASIC und andererseits flexibler als der BASIC-Befehl selbst. Die Programmierpsrache C vereint also Einfachheit und Flexibilität. Das zeigt sich besonders bei der Übergabe der Daten an die Funktion. Hier ist ein Listing, das ein Window öffnet, einige Zeit wartet und dieses dann wieder schließt:

```
/*****************************************
 *                                       *
 * Programm: Window öffnen mit NewWindow *
 * ===================================== *
 *                                       *
 * Autor:  Datum:      Kommentar:        *
 * ------  ----------  ----------        *
 * Wgb     14.10.1988                    *
 *                                       *
 *                                       *
 *****************************************/
#include <intuition/intuition.h>
struct IntuitionBase *OpenLibrary(), *IntuitionBase;
main()
   {
   struct Window        *UnserWindow, *OpenWindow();
   struct NewWindow     *WinDat;
   if (!(IntuitionBase =
                  OpenLibrary("intuition.library", 0L)))
      {
      printf("Keine Intuition-Library gefunden!\n");
      exit(FALSE);
      }
   WinDat.LeftEdge    = 160;
   WinDat.TopEdge     =  50;
   WinDat.Width       = 320;
   WinDat.Height      = 200;
   WinDat.DetailPen   =   0;
   WinDat.BlockPen    =   1;
   WinDat.IDCMPFlags  = NULL;
```

```
WinDat.Flags       = WINDOWDEPTH | WINDOWSIZING |
                     WINDOWDRAG  | WINDOWCLOSE  |
                     SMART_REFRESH;
WinDat.FirstGadget = NULL;
WinDat.CheckMark   = NULL;
WinDat.Title       = (UBYTE *)"Intuition Test";
WinDat.Screen      = NULL;
WinDat.BitMap      = NULL;
WinDat.MinWidth    = 100;
WinDat.MinHeight   =  50;
WinDat.MaxWidth    = 640;
WinDat.MaxHeight   = 256;
WinDat.Type        = WBENCHSCREEN;

UnserWindow = OpenWindow(&WinDat);

Delay(200L);

CloseWindow(UnserWindow);

CloseLibrary(IntuitionBase);
}
```

Listing 6.2.9: Ein Window öffnen

Programmbeschreibung

Nachdem zu Beginn der Funktion main() wieder die OpenLibray()-Funktion und die Variable IntuitionBase definiert wurden, benötigen wir zwei weitere Platzhalter. Einmal den Zeiger UnserWindow, der später auf das Window zeigen wird, und die Struktur Window mit dem Namen WinDat, in der die Einschalt-Daten des Windows liegen. Damit sind alle Einstellungen gemeint, die beim Öffnen gemacht werden können.

Hier finden wir neben den bekannten Werten, die über den BASIC-Befehl vertraut sind, auch noch diejenigen wieder, die erst über die Library-Ansteuerung zugänglich und auch nötig wurden. Nun aber haben wir Zugang zu allen Einstellungen, also auch den internen, die nur für die Programmierung gebraucht werden. Da wir hier aber programmieren, müssen wir die Parameter auch einstellen können. Das machen wir!

Zuerst wird mit Top- und Left-Edge die Position des neu geöffneten Windows bestimmt. Auch die Höhe und Breite kann mit Height und Width eingestellt werden. Beachten Sie, daß hier große Differenzen zwischen HiRes und LoRes bestehen. Man sollte also immer sicher sein, auf welchem Screen man sein Window öffnet. Aber weil unseres auf der Workbench erscheinen wird, haben wir gar keine Probleme.

Dann kommen wir zur Farbeinstellung. DetailPen legt die Schrift- und Block-Pen die Hintergrund-Farbe fest. Beide wurden in AmigaBASIC gar nicht und von unserem Unterprogramm automatisch eingestellt. Das vereinfachte die Parameterübergabe, weil diese Werte ja auch selten gebraucht werden. Aber testen Sie ruhig einige Kombinationen.

Jetzt sind die Flaggen an der Reihe. IDCMP steht für den Nachrichtenport, der uns aber weniger interessieren soll. Wichtig sind die normalen Flaggen, mit denen man bestimmt, welche System-Gadgets das Window besitzen soll. Wählen Sie hier, was für den Benutzer am sinnvollsten ist. Der Refresh-Modus SMART sollte erhalten bleiben, weil er den Inhalt des Windows vor unerwünschtem Löschen weitestgehend schützt.

Die ersten Einträge in die Datentabelle waren durchweg Werte. Jetzt haben wir es mit einem Zeiger zu tun. FirstGadget beinhaltet die Adresse des ersten Gadgets, das in diesem Window vom Benutzer definiert wurde. Diese Verkettung von Window und Gadgets wurde in dem Programm zur File-Request-Box benutzt. Erst definiert man die Gadget-Daten und setzt den Zeiger auf das erste in das Window ein. Schon sind die Gadgets mit dem Window verbunden.

Bei CheckMark handelt es sich wiederum um einen Zeiger, der diesmal auf eine Grafik zeigt. Diese wird bei der Menüausgabe zur Darstellung des Hakens benutzt. Ein sehr selten genutztes Einstellungsmerkmal.

Mit dem Eintrag Titel haben wir wieder festen Boden unter den Füßen. Jedes Window bekommt einen Titel, und diesen legen wir hier ab. Allerdings wird auch hier wieder nur ein Zeiger auf den eigentlichen Text gespeichert. Aber darüber müssen wir uns gar keine Gedanken machen, denn der C-Compiler erledigt dieser Arbeiten für uns.

Ein Window muß nicht, wie wir schon im nächsten Beispiel sehen werden, auf der Workbench geöffnet werden. Sie haben sicherlich schon bei anderen

Programmen bemerkt, daß diese sich eigene Screens öffnen und auch darauf neue Windows aufmachen. Das wird nicht unbedingt nur aus dem Grund getan, daß man keine Kollision mit den Workbench-Windows haben möchte. Schließlich ist der Speicherverbrauch dagegen ein schwerwiegendes Argument. Viel entscheidender ist der Grund, daß die Workbench-Screen eine festgelegte Auflösung hat. Der Amiga bietet aber viel mehr als nur eine Auflösung, und die der Workbench eignet sich nicht für jedes Problem. Kurzum, in dieser Variablen der NewWindow-Struktur steht ein Zeiger auf eine Custom-Screen, wenn das Window nicht auf der Workbench-Screen geöffnet werden soll.

Neben einer eigenen Screen ist es dann auch noch möglich, seinen eigenen Grafik-Speicher für das Window zu verwalten. Mit dieser Methode kann man sehr interessante grafische Gebilde unterstützen, die sich sonst nicht so einfach realisieren lassen. Aber dieses Thema sprengt nun wirklich den Rahmen des Buches. Im Supergrafik-Buch von DATA BECKER finden Sie dazu einige Beispiele sowohl in BASIC als auch in C.

Von der Workbench kennen Sie das bestimmt! Die Windows lassen sich zwar in der Größe verändern, bei bestimmten Grenzen geht aber nichts mehr. Ein Window läßt sich also nicht beliebig klein machen. Wie klein, bestimmen nun die MinWidth- und MinHeight-Werte in der Struktur. Genauso kann auch die maximale Ausdehnung eines Windows bestimmt werden, damit z.B. nicht unendlich viel Speicher verbraucht wird. Aber bedenken Sie auch, daß so eine Begrenzung ärgerlich sein kann. Das beste Beispiel dafür ist das AmigaBASIC-Output-Window, das auf einen amerikanischen Bildschirm eingestellt ist. Die Auflösung der Workbench-Screen kann jederzeit geändert werden! Das muß man als Programmierer immer vor Augen haben. Tragen Sie hier entweder die gewünschten Maxi- und Minimalwerte ein oder eine Null. Diese verbietet dann das Verändern der Größe, weswegen auch das WINDOWSIZING-Flag entfernt werden kann. Möchte man, daß jeweils die größt- und kleinstmöglichen Werte zugelassen sind, so ist einfach -1 als Wert anzugeben.

Als letzten Parameter unserer NewWindow-Struktur haben wir den Eintrag Type, mit dem bestimmt wird, auf welcher Screen das Window geöffnet werden soll. Dies zeigt für Intuition nur an, ob es nachsehen muß, ob in Screen ein Screen-Zeiger eingetragen ist. Setzen Sie hier WBENCHSCREEN ein, wenn es sich um ein Window handelt, das auf der Workbench geöffnet werden soll. In diesem Fall muß natürlich kein Zeiger in der Screen-Variablen eingetragen werden.

Haben Sie aber eine eigene Screen vorher aufgemacht, kann der Zeiger eingetragen und die Flagge CUSTOMSCREEN gesetzt werden. Fertig!

Ein bißchen Nachrichtenübermittlung

Dieses Window-Programm läßt sich nun auf verschiedene Weise erweitern. Es bietet sich an dieser Stelle an, einige Benutzer-Nachrichten zur Kommunikation zu empfangen und auszuwerten. Dazu erweitern wir das Window-Programm so, daß es Nachrichten über seinen Datenport empfangen kann. Sie werden fragen, wozu es wichtig ist, daß unser Programm Nachrichten-Sendungen sehen kann. Aber so war das nicht gemeint. Es soll vielmehr Nachrichen vom Benutzer empfangen. So z.B. Tastendrücke oder Mausklicks.

Intuition bietet für die Kommunikation zwischen Benutzer und Programm einen Datenkanal, auf dem alle gewünschten Nachrichten übertragen werden. Hier sehen Sie schon das fertige Listing, in dem die nötigen Teile eingefügt wurden:

```
/************************************
 *                                  *
 * Programm: Nachrichten empfangen  *
 * ================================ *
 *                                  *
 * Autor:  Datum:      Kommentar:   *
 * ------  ----------  ----------   *
 * Wgb     15.10.1988               *
 *                                  *
 ************************************/
#include <intuition/intuition.h>
struct IntuitionBase *OpenLibrary(), *IntuitionBase;
struct IntuiMessage  *GetMsg();

main()
   {
   struct Window    *UnserWindow, *OpenWindow();
   struct NewWindow  WinDat;
   struct IntuiMessage *Message;
   ULONG MessageClass;
   BYTE  MessageCode;
   if (!(IntuitionBase =
            OpenLibrary("intuition.library", 0L)))
      {
      printf("Keine Intuition-Library gefunden!\n");
```

```
    exit(FALSE);
    }
WinDat.LeftEdge     = 160;
WinDat.TopEdge      =  50;
WinDat.Width        = 320;
WinDat.Height       = 200;
WinDat.DetailPen    =   0;
WinDat.BlockPen     =   1;
WinDat.IDCMPFlags   = VANILLAKEY  | CLOSEWINDOW;
WinDat.Flags        = WINDOWDEPTH | WINDOWSIZING |
                      WINDOWDRAG  | WINDOWCLOSE  |
                      SMART_REFRESH;
WinDat.FirstGadget  = NULL;
WinDat.CheckMark    = NULL;
WinDat.Title        = (UBYTE *)"Intuition Test";
WinDat.Screen       = NULL;
WinDat.BitMap       = NULL;
WinDat.MinWidth     = 100;
WinDat.MinHeight    =  50;
WinDat.MaxWidth     = 640;
WinDat.MaxHeight    = 256;
WinDat.Type         = WBENCHSCREEN;
UnserWindow = OpenWindow(&WinDat);
FOREVER
   {
   if ((Message =
            GetMsg(UnserWindow->UserPort)) == NULL)
      {
      Wait(1L << UnserWindow->UserPort->mp_SigBit);
      continue;
      }
   /* Auswertung */
   MessageClass = Message->Class;
   MessageCode  = Message->Code;
   ReplyMsg(Message);
   switch(MessageClass)
      {
      case CLOSEWINDOW : CloseWindow(UnserWindow);
                         CloseLibrary(IntuitionBase);
                         exit(TRUE);
                         break;
      case VANILLAKEY  : printf("%c\n", MessageCode);
                         break;
      }
   }
}
```

Listing 6.2.10: Kommunikation Benutzer-Programm

Programmbeschreibung

Ergänzt wurde an diesem Programm hauptsächlich die Endlos-Schleife am Ende des Listings, die nach dem Öffnen des Windows seinen Nachrichtenport ständig auf anliegende Nachrichten hin überprüft. Nur wenn eine Nachricht anliegt, wird zur Auswertung verzweigt, in der an dieser Stelle zwei verschiedene Möglichkeiten überprüft werden. Zum einen, und auch als wichtigstes, das Schließen des Windows, das zur Folge hat, daß sowohl die Library als auch das Window geschlossen werden. Das Programm beendet dann seine Arbeit.

Der zweite Nachrichten-Typ ist ein Tastendruck. Er wird hier nicht weiter untersucht, sondern gleich im AmigaDOS-Window ausgegeben. Später, wenn das Programm verschiedene Funktionen zur Verfügung stellt, werden wir abfragen, um welche Taste es sich gehandelt hat, und entsprechend darauf reagieren.

7. PC

Wenden wir uns nun dem Thema PC zu. Mit den PC, auch Personal Computer genannt, ist das so eine Sache. Erfunden wurden Sie vor "vielen" Jahren vom Computerriesen IBM - einer Firma, die es gewohnt ist, stark überhöhte Preise zu veranschlagen. Offensichtlich konnte (und kann) sich IBM solche Preise erlauben. Da es aber auch Leute gegeben hat, die nicht einfach 10000,- DM für einen Computer ausgeben konnten, der nicht viel besser war, als Geräte mit ähnlichen Leistungsmerkmalen anderer Hersteller, und die andererseits auch gerne einen Rechner hätten, auf dem die ohne Zweifel gute IBM-Software funktioniert, drängten viele große und kleine Computerhersteller vom Weltkonzern bis zur Garagenfirma auf den PC-Markt. Sie entwickelten Geräte, auf denen die gleichen Programme funktionierten wie auf den Orginal IBMs. Auch in der Hardware waren diese Nachbauten weitestgehend kompatibel. Das Betriebssystem hatte IBM von der Firma Microsoft entwickeln lassen, die es auch an die Fremdhersteller verkaufte. Durch einen immer härter werdenden Preiskampf kam es dazu, daß man heute PC für unter 1000,- DM kaufen kann. Aufgrund der schnell fortschreitenden Technik wurde es sogar möglich, einen PC auf einem anderen Gerät (z.B. einem Amiga) zu emulieren, d.h. nachzuahmen. Dabei kann man sich entweder an eine langsame und nicht unbedingt besonders zuverlässige Software-Realisierung oder an eine recht schnelle - aber auch etwas teurere - Hardware halten. Bei der Hardware-Version wird eine komplette PC-Platine an den Amiga angeschlossen. Doch dazu später.

MS-DOS ist die Bezeichnung für das auf PC übliche Betriebssystem. Es besteht, anschaulich gesagt, aus zwei Teilen, dem System-Teil, mit dem wir als Anwender nicht direkt in Berührung kommen, und der Benutzerschnittstelle, der sogenannten Shell, die uns in diesem Kapitel noch beschäftigen wird. Sie weist eine starke Ähnlichkeit zum CLI auf, was uns Grund zur Annahme gibt, daß sich die Entwickler bei Commodore ein paar Ideen "geborgt" haben, um sie in einem besseren System zu verwerten. MS-DOS hat aber auch einige Vorteile gegenüber dem CLI. Hier wäre z.B. die Ordnung und Einfachheit zu nennen. Um den MS-DOS-Teil richtig verstehen zu können, ist es günstig, sich etwas im CLI auszukennen.

Lange Rede - kurzer Sinn, kommen wir zur Installation des PC.

7.1 Die Software-Lösung

Beginnen wir mit der Software-Variante. Sie besteht aus zwei 3½-Zoll-Disketten, auf denen sich neben dem eigentlichen Betriebssystem MS-DOS noch ein Programm befindet, das die Hardware eines PC einschließlich des Prozessors emuliert. Dieser Emulator befindet sich auf der sogenannten Transformer-Disk. Er ist nötig, da Personal Computer mit anderen Prozessoren arbeiten als der Amiga. Der Emulator simuliert nun jeden einzelnen Prozessor-Befehl des laufenden PC-Programms. Aus diesem Grund arbeitet der nachgeahmte PC auch nicht so schnell wie ein richtiger. Für kleine Anwendungen reicht die Geschwindigkeit jedoch aus.

Die Tatsache, daß fast alles nur emuliert wird, läßt die berechtigte Vermutung aufkommen, daß es Programme geben kann, die nicht richtig funktionieren. In dieser Hinsicht waren wir jedoch angenehm überrascht. Programme, die keine Grafik verwendeten, liefen einwandfrei. Bei einigen Testroutinen, die wir selbst in Maschinensprache geschrieben hatten, zeigten sich jedoch Abweichungen zu einem normalen PC. Eigentlich fielen sie aber alle in den Bereich der Grafikdarstellung und der Umschaltung in eine andere Auflösung, die gar nicht geklappt hat.

Neben dem eigentlichen Emulator gibt es noch ein Programm, das eine ähnliche Funktion hat, wie Preferences. Es trägt den bezeichnenden Namen SetAtPrefs. Mit ihm können einige Einstellungen für den Emulator geändert werden. Starten Sie dieses Programm jetzt bitte.

In dem Fenster, das sich nun öffnet, können Sie mit den Cursortasten die einzelnen Funktionen anfahren. Drücken Sie [Return], gelangen Sie in ein entsprechendes Untermenü, das Sie dann mit [Esc] wieder verlassen können. Auch das Hauptmenü wird mit dieser Taste beendet. Nach einem Druck auf [Esc] werden automatisch alle Änderungen abgespeichert. Folgende Untermenüs werden angeboten:

Menü	Aufgabenbereich
Disk Menu	Laufwerke einteilen
Video Menu	Modus und Cursor einstellen
Memory Menu	Speicher zuteilen
Attachments Menu	Tastaturwiederholung einstellen
Mono Menu	Farben für Monochrombetrieb einstellen

Wir werden uns nun nach und nach durch alle oben genannten Menüs durcharbeiten. Beginnen wollen wir mit dem "Disk Menu". Rufen wir es auf, erscheint eine neue Bildschirmseite, auf der man MS-DOS die vorhandenen Laufwerke zuteilen kann. Das wollen wir jetzt auch tun. Alle Einstellungen werden ohne Maus vorgenommen - wahrscheinlich, um den Anwender schon auf das vorzubereiten, was ihn erwartet, wenn er richtig mit MS-DOS einsteigt.

Mit [↑] und [↓] wählen wir den Unterpunkt an, den wir ändern wollen. Vier solcher Unterpunkte stehen zur Auswahl. Jeder von ihnen vertritt ein MS-DOS-Laufwerk. MS-DOS teilt jedem Laufwerk, jeder Festplatte und jeder RAM-Disk eine Buchstabenkennung zu. So ist z.B. A: das erste und B: das zweite Laufwerk. Um ein MS-DOS-Laufwerk auf eine Amiga-Floppy zu fixieren, muß der Cursor in die Zeile gebracht werden, die mit der besagten Buchstabenkennung beginnt. Drückt man nun [Return], wandert ein Klammernpaar von einer Möglichkeit zur nächsten. Bei Laufwerk A: kann man "DF0:", "DF1:", "DF2:" und "DF3" anwählen. Bei allen anderen wird zusätzlich noch "Not used" angeboten. Mit "Not used" kann man festlegen, daß man der PC-Emulation dieses Laufwerk nicht gönnen will. Man sollte jedoch alle verfügbaren Laufwerke bereitstellen, da die Emulation ohnehin nicht im Multitasking-Betrieb eingesetzt werden kann und für den Amiga daher keine Laufwerke übrigbleiben müssen.

Nachdem wir das Menü mit [Esc] verlassen haben, stürzen wir uns schon ins nächste. Über das "Video Menu" können wir wählen, welche Darstellungsmodi wir verwenden möchten. Dabei stellen wir einfach alles auf "Yes". Für "Starting video mode" können wir entweder "MONO" oder "COLOR" angeben - ein Unterschied zwischen den beiden Varianten konnte von uns allerdings nicht festgestellt werden. Bleibt nur noch "Cursor type". Hier bieten sich "UNDERLINE" und "BLOCK" an. Wählt man "UNDERLINE", wird das Zeichen an Cursorposition unterstrichen, ansonsten wird es invertiert. Die Einstellung, die hier vorgenommen wird, ist nicht besonders wichtig, da sie ohnehin nur für den Start von MS-DOS gilt. Fast jedes Programm stellt sich den Cursor nämlich selbst ein.

Weiter geht es im "Memory Menu". Es gibt uns die Möglichkeit, festzulegen, wieviel Speicher die Emulation erhalten soll. Dabei müssen zwei Dinge eingestellt werden: ein Minimal- und ein Maximalwert. Das Programm versucht dann, die als Maximum angegebene Speichermenge zu belegen. Steht nicht genug zusammenhängender Speicher zur Verfügung, gibt es sich auch mit

weniger zufrieden, sofern es noch mehr ist, als beim Minimalwert angegeben ist. Andernfalls bricht der Emulator ab. Dieses Menü ist, unserer unwesentlichen Meinung nach, völliger Unsinn. Da es aber nun mal da ist, müssen wir auch ein paar sinnvolle Werte einstellen. Dazu bringen wird den Cursor zunächst auf die Minimumzeile. Dann drücken wir so lange [F6], bis dort der Wert 64 steht. Den Maximalwert stellen wir, nachdem wir ihn mit [↓] angewählt haben, mit [F5] auf 640. Dann verlassen wir auch dieses Menü.

Nächster Punkt der Tagesordnung ist das "Attachments Menu". Es ermöglicht uns, die Geschwindigkeit und die Verzögerung der Tastenwiederholung einzustellen.

Der Wert hinter "Keybord delay (seconds) before repeat" gibt die Zeit in Sekunden an, die der PC wartet, bis er mit der Tastenwiederholung beginnt. Wie viele Zeichen er dann pro Sekunde ausgibt, stellen wir mit "Keyboard repeat (chars/sec)" in Zeichen pro Sekunde ein. Möchten wir auf die Tastenwiederholung ganz verzichten, müssen wir OFF aktivieren.

Das letzte der Menüs gestattet es uns, die Farben einzustellen, mit denen man im Monochrom-Modus arbeiten möchte. Da es uns nicht gelungen ist, in den Farbbetrieb umzuschalten, gehen wir davon aus, daß diese Werte immer gültig sind. Mit den Cursortasten wählen wir nun zunächst die Hintergrundfarbe (Background) an. Nun können wir mit den Funktionstasten [F1] bis [F6] deren Farbzusammensetzung ändern.

Farbe	erhöhen	erniedrigen
ROT	[F1]	[F2]
GRÜN	[F3]	[F4]
BLAU	[F5]	[F6]

Dann ist die Zeichenfarbe (Character) an der Reihe. Auch hier stellen wir uns unsere Lieblingsfarbe zusammen. Nun haben wir alle Einstellungen vorgenommen und können SetAtPrefs mit [Esc] verlassen.

Endlich ist es soweit, wir können den Transformer (PC-Emulator) starten. Nach kurzer Ladezeit, wechselt die Bildschirmfarbe über das typische Reset-Weiß zur von uns eingestellten Hintergrundfarbe (z.B. schwarz). Wir werden mit

```
Please insert a DOS diskette in drive A:
press Return to start the DOS
or
press Esc to return to the Amiga DOS
```

aufgefordet, eine DOS-Diskette einzulegen und [Return] zu drücken. Dieser Aufforderung kommen wir nach, indem wir die zweite Diskette, auf der sich das eigentliche MS-DOS befindet, in das Laufwerk einlegen, das wir in "SetAtPrefs" als A: deklariert haben. Nach einem Druck auf die [Return]-Taste, wird MS-DOS geladen. Erstaunlicherweise stürzt der Emulator jedoch gelegentlich ab, wenn man Fast-RAM installiert hat. Es ist daher ratsam, dieses abzuschalten (siehe auch Kapitel 2.9).

Wer die Möglichkeit hat, eine andere DOS-Version zu bekommen, die besser ist, als die, die mit dem Emulator geliefert wird, kann diese auch verwenden.

Wenn Sie kein Interesse am SideCar oder der PC-Karte haben, können Sie schon mal zu Kapitel 7.3 vorblättern, in dem wir Sie in die Arbeit mit MS-DOS einführen. Allen anderen stellen wir nun die PC-Hardware vor.

7.2 Die Hardware-Lösung: SideCar oder PC-Karte

Wem die Software zu langsam ist, der sollte sich mit dem Gedanken vertraut machen, das SideCar mit einem passenden Adapter oder eine PC-Karte zu installieren. Die PC-Karte muß in das Tuning-Gehäuse von Comptec eingebaut werden, da der 500er in der Grundausstattung keine 2000er-Steckplätze hat. Dabei gibt es allerdings ein kleines Problem: Im Tuning-Gehäuse gibt es keinen Schacht, in den man das mit der PC-Karte gelieferte Laufwerk einbauen könnte. Diese Variante ist daher eher für Bastler interessant, die das Laufwerk extern in einem eigenen Gehäuse unterbringen können. Allen anderen empfehlen wir das SideCar, das zur Zeit bei einer Versandfirma (DATA 2000) für läppische 499,- Mark bestellt werden kann. Aber, wie gesagt: Ein Adapter ist notwendig.

Sollte Ihrem SideCar oder Ihrer PC-Karte keine Betriebs-Software beiliegen, wenden Sie sich bitte an Ihren Fachhändler. Er kann Ihnen diese Programme zur Verfügung stellen. Sie befinden sich z.B. auf älteren Workbench-Disketten zum Amiga 2000.

Dieses Software-Paket erlaubt es, sowohl Monochrom- als auch Farbgrafik des PC auf dem Amiga-Monitor in einem Fenster oder auf einem eigenem Screen darzustellen. Wechselt ein Programm allerdings den Ausgabe-Modus, müssen Sie den Screen bzw. das Fenster anwählen. Mit diesem Manko kann man jedoch ganz gut leben, da Programme im allgemeinen einen Modus durchgehend verwenden.

Befinden Sie sich in der MS-DOS-SHELL, können Sie den Modus mit dem Befehl MODE ändern. Dabei schaltet

```
MODE CO80
```

die Farb- und

```
MODE MONO
```

die Monochromgrafik ein.

Die Text- und Grafikdarstellung wirkt manchmal etwas langsam. Dies liegt aber nicht etwa daran, daß der PC langsam arbeitet, sondern daß die Grafik, die auf der PC-Seite schon vorliegt, erst auf die Amiga-Seite übertragen werden muß. Da der PC davon nichts merkt (er weiß nämlich gar nicht, daß er nur ein Amiga-Sklave ist) wird seine Arbeitsgeschwindigkeit auch nicht gedrosselt.

7.3 Arbeiten mit dem PC

Installiert haben wir ihn nun - unseren kleinen PC. Auch wissen wir, wie wir ihn aktivieren. Alles schön und gut, aber wie wird er bedient? Dieser Frage wollen wir nun ansatzweise auf den Grund gehen. Es ist jedoch nicht unsere Absicht, Ihnen alles zu erzählen, was es zu erzählen gibt. Dazu gibt es schließlich genug PC-Bücher. Wir möchten Ihnen vielmehr den Umstieg vom Amiga zum PC erleichtern, der nicht immer ganz einfach ist - besonders, wenn man schon von Workbench, Maus, Symbolen und Menüs verwöhnt ist. Ein PC besitzt in der Grundaustattung keine Benutzeroberfläche wie der Amiga. Man muß sich mit einem System anfreunden, das dem CLI recht ähnlich ist. Identisch sind MS-DOS, das Betriebssytem des PC, und AmigaDOS bzw. CLI aber auch nicht. Wie Sie noch sehen werden, gibt es doch den einen

oder anderen Unterschied. Nicht verschweigen wollen wir, daß man auf einem normalen PC kein Multitasking einsetzen kann. Auch wenn der PC nicht so tolle Möglichkeiten bietet wie der Amiga, hat er trotzdem seine Vorteile.

Vom CLI zu MS-DOS

MS-DOS spielt für PC-Anwender eine große Rolle. Mit seiner Hilfe kann man Programme starten, Disketten formatieren und BootDisketten anlegen. Auch kann man es zum Kopieren von Dateien und Disketten einsetzen und und und... Leider ist der Umgang mit MS-DOS für reine Anwender oft eine unangenehme Plage. Hat man das System aber erst einmal richtig durchschaut, kann man rundrum zufrieden sein. Nehmen wir uns also ein Herz und beginnen.

Nach dem Booten des Systems sieht man nach kurzer Zeit den MS-DOS-Prompt. Er sieht üblicherweise so aus:

```
A>
```

Im CLI gab es auch einen Prompt, der sich aus der Tasknummer des CLI und dem ">"-Zeichen zusammensetzte. Da MS-DOS nicht multitaskingfähig ist, muß das "A" eine andere Bedeutung haben. Es handelt sich hierbei um das aktuelle Laufwerk. Möchte man auf ein anderes Laufwerk zugreifen, muß man dessen Buchstabenkennung, gefolgt von einem Doppelpunkt, eingeben, z.B.

```
B:
```

Bei der nächsten Ausgabe des Prompts zeigt dieser

```
B>
```

Besitzen Sie kein PC-Zweitlaufwerk, können Sie natürlich nicht auf B: umstellen. Auch andere Speichermedien werden über das oben genannte Verfahren eingestellt. C: steht z.B. für die Festplatte. Doch auch eine RAM-Disk kann man unter diesem Buchstaben installieren. Stellen Sie jetzt auf alle Fälle wieder auf A: um.

```
A:
```

Beim CLI wurden alle Befehle von Diskette geladen. Bei MS-DOS sieht die Sache etwas anders aus. Man unterscheidet hier zwischen internen und externen Befehlen. Interne Befehle sind immer aufrufbar, externe müssen erst nachgeladen werden.

Ein residenter (= interner) Befehl ist CD. Dieses Kommando hat fast die gleiche Aufgabe wie der CD-Befehl des CLI. Allerdings kann man in MS-DOS CD nur zur Wahl des Unterverzeichnisses verwenden. Für jedes angeschlossene Laufwerk wird ein eigener Pfad gespeichert. Dieser Pfad bleibt erhalten, wenn man auf ein anderes Gerät umschaltet. Schaltet man wieder zurück, ist er wieder aktiv. Im Gegensatz zum CLI arbeitet MS-DOS mit den Kennungen der Laufwerke und nicht mit den Diskettennamen. Man muß also nicht ständig CD eingeben.

Um den Inhalt eines Verzeichnisses auszugeben, verwendet man den Befehl DIR. Er gibt, wenn man keine besonderen Parameter angibt, das Verzeichnis aus, auf das der Pfad des aktuellen Laufwerks zeigt. Möchte man das Verzeichnis eines anderen Laufwerks sehen, muß man dessen Kennung angeben. Hat man für dieses Laufwerk mit CD ein Unterverzeichnis gewählt, dann erhält man dessen Inhalt.

Unterverzeichnisse werden mit MD angelegt und mit RD gelöscht. Löschen kann man sie allerdings nur, wenn sie auch leer sind. Im Gegensatz zum CLI unterscheidet man beim Löschen zwischen Dateien, die mit Del gelöscht werden, und Verzeichnissen, die, wie schon gesagt, mit RD entfernt werden. Es ist also vom MS-DOS aus nicht möglich, einen Directory-Baum komplett zu löschen.

Disketten werden ebenfalls von MS-DOS formatiert. Dazu gibt es den Befehl Format. Er ist sehr leicht zu bedienen: Einfach MS-DOS-Diskette einlegen,

```
FORMAT A:
```

eingeben und den Anweisungen des Systems folgen. Will man von der zu formatierenden Diskette booten, muß man

```
FORMAT A: /S
```

verwenden. Dadurch wird automatisch das Betriebssystem übertragen. Nachträglich ist es nämlich nicht ohne weiteres möglich, eine Diskette bootfähig zu

machen. FORMAT kann wie auch alle anderen Befehle, die wir Ihnen noch vorstellen werden, viel mehr, als wir beschreiben. Wir wollen Ihnen natürlich keine Informationen vorenthalten, können aber auch nicht alles, was es zu MS-DOS zu berichten gibt, in diesem Buch abdrucken. Wer Interesse an MS-DOS gefunden hat, den können wir auf diverse Literatur zu diesem Thema verweisen.

Doch zurück zum Thema! Möchte man ein Programm aufrufen, muß man einfach dessen Namen eingeben. Dabei gibt es allerdings wieder eine Abweichung zu AmigaDOS. In MS-DOS bestehen Filenamen nämlich aus zwei Teilen: einem maximal achtstelligen Namen und einer maximal dreistelligen Extension, die angibt, um welchen Datentyp es sich handelt. Ein Anwendungsprogramm gibt den Daten, die von ihm bearbeitet werden, eine eigene Kennung. Dadurch kann man alle Files, die von diesem Programm erstellt wurden, leicht finden und gezielt bearbeiten.

Für MS-DOS sind drei Endungen reserviert. COM und EXE sind für ausführbare Programme zuständig. Zwischen ihnen besteht nur ein kleiner, für uns nicht wichtiger Unterschied. BAT ist für Batch-Files zuständig, die ähnlich arbeiten wie die des Amiga. Allerdings muß man keinen Befehl eingeben, um sie aufzurufen.

Sehen Sie sich bitte einmal das folgende Inhaltsverzeichnis an:

```
 Datenträger in Laufwerk A hat keine Bezeichnung
 Inhaltsverzeichnis von  A:\

PROGRAMM COM     32324  01.01.88  14:10
.
.
.
FILE     TXT      1212  12.04.88  06:50
BATCH    BAT        40  19.08.88  13:07
       43 Datei(en)      46080 Bytes frei
```

Der ersten Zeile entnehmen wir, daß die Diskette, deren Verzeichnis wir ausgegeben haben, keinen Namen hat. Danach wird angezeigt, um welches Inhaltsverzeichnis es sich eigentlich handelt. A:\ vertritt das oberste Verzeichnis von A:

Dann kommen die Files. Nach dem eigentlichen Namen folgen die Extension, die Länge und das Erstellungsdatum. Die drei Pünktchen haben wir stellvertretend für die restlichen Einträge eingesetzt.

Schließlich werden wir noch über die Anzahl der Dateien informiert. Auch erfahren wir, wie viele Bytes auf der Diskette noch verfügbar sind.

Spätestens jetzt sollten wir eine Kopie der MS-DOS-System-Diskette anlegen. Dazu verwenden wir den Befehl DiskCopy. Als Parameter geben wir das Quell- und Ziellaufwerk an.

```
DiskCopy a: b:
```

Ab jetzt werden wir nur noch mit der Kopie arbeiten, da wir zur Demonstration der wichtigsten Kommandos einige Veränderungen vornehmen müssen.

Zuvor wollen wir allerdings noch zwei weitere Befehle vorstellen. Da wäre zunächst REN. Alternativ kann man auch RENAME verwenden. Die Funktion des Befehls ist aus dessen Namen bereits erkennbar: Ein File wird umbenannt.

Sehr wichtig ist auch Copy. Mit Copy können einzelne Dateien und auch ganze Gruppen kopiert werden. Legen Sie bitte die Arbeitskopie der System-Diskette in A:, und machen Sie die folgenden Beispiele mit.

Zunächst schaffen wir uns eine Testumgebung. Dazu stellen wir zuerst auf A: um.

```
A:
```

Um sicherzustellen, daß wir auch im Hauptverzeichnis sind, geben wir noch

```
CD \
```

ein. Der Backslash bedeutet, wenn er allein oder vor einem Datei- oder Verzeichnisnamen steht, daß das oberste Verzeichnis angesprochen werden soll.

Wenn Sie nun

```
CD
```

aufrufen, erhalten Sie die Ausgabe:

```
A:\
```

Doch was passiert bei CD A:? Hier wird wieder der Pfad angezeigt. Für einen CLI-Anwender mag dies verblüffend sein, in MS-DOS ist das jedoch ganz normal. Hätten Sie CD B: eingegeben, hätte Ihnen CD den Pfad geliefert, der für Laufwerk B: eingestellt ist.

Nun legen wir ein Verzeichnis an,

```
MD TEST
```

in das wir dann auch gleich verzweigen, um uns anschließend das Directory des neu angelegten Verzeichnisses ausgeben zu lassen.

```
CD TEST
DIR
```

Das Verzeichnis sieht in etwa so aus:

```
 Volume in Laufwerk A hat keinen Namen
 Verzeichnis von  A:\TEST

             <DIR>       1-01-80  12:01a
             <DIR>       1-01-80  12:01a
        2 Datei(en)      39936 Bytes frei
```

Doch was haben die Einträge

```
             <DIR>       1-01-80  12:01a
             <DIR>       1-01-80  12:01a
```

in unserem Directory zu suchen? Die haben wir doch gar nicht angelegt! Wie an der Kennung <DIR> zu erkennen ist, handelt es sich um Directories. In der Tat hat MS-DOS hier zwei Verzeichnisse installiert, die allerdings nur zu Hilfszwecken dienen. Mit CD .. gelangt man nämlich, ähnlich dem CD / des CLI ein Verzeichnis zurück.

Im Verzeichnis TEST legen wir nun ein weiteres Unterdirectory an, in das wir, nachdem wir uns das Inhaltsverzeichnis nochmals angesehen haben, verzweigen:

```
MD TURBO
DIR
CD TURBO
```

Der Aufruf von DIR lieferte uns nun die folgende Ausgabe:

```
 Volume in Laufwerk A hat keinen Namen
 Verzeichnis von  A:\TEST

.            <DIR>       1-01-80  12:01a
..           <DIR>       1-01-80  12:01a
TURBO        <DIR>       1-01-80  12:03a
        3 Datei(en)      37888 Bytes frei
```

So, jetzt geht es erst richtig los. Wir kopieren eine Datei aus dem Hauptverzeichnis in das nun aktuelle TURBO-Directoy. Als Beispiel nehmen wir die Datei COMMAND, die mit der Extension COM behaftet ist. Doch da fangen die Probleme schon an. Es stellt sich nämlich die Frage, wie die Extension anzugeben ist. Dazu gibt es ein allgemeines Verfahren. Man gibt den Dateinamen, gefolgt von einem Punkt (.) und der Extension, an. Dann geht alles klar. Filenamen unter MS-DOS dürfen übrigens keine Leerstellen enthalten und sind auch sonst einigen Beschränkungen unterworfen, die jedoch von Version zu Version variieren können. Doch zurück zu COPY:

```
COPY \COMMAND.COM MEINFILE.MLA
```

Man beachte den Backslash vor COMMAND.COM, der COPY mitteilt, daß die zu kopierende Datei im Hauptverzeichnis zu finden ist.

Alternativ hätte man auch

```
COPY ..\..\COMMAND.COM MEINFILE.MLA
```

oder

```
COPY \COMMAND.COM \TEST\TURBO\MEINFILE.MLA
```

schreiben können.

Nächster Schritt ist das Kopieren ganzer Filegruppen. Dazu gibt es, ähnlich wie im CLI, Joker. Das Fragezeichen vertritt ein einzelnes Zeichen, das Sternchen beliebig viele. Möchte man alle Dateien ansprechen, reicht es aber nicht

aus, einfach ein Sternchen (*) zu schreiben, da man immer noch die Extension angeben muß. Richtig wäre es, sowohl für den Namen als auch für die Extension ein Sternchen einzugeben (*.*).

```
COPY \*.SYS ..
```

kopiert alle Files mit der Erweiterung SYS in das Verzeichnis TEST, das wir mit den zwei Pünktchen angeben konnten, da es zur Zeit das nächst höhere Verzeichnis ist.

Daß man mit COPY auch Dateien von einem Laufwerk auf ein anderes kopieren kann, versteht sich von selbst. Hier ein Beispiel:

```
COPY A:\*.TXT B:\
```

kopiert alle Files mit der Kennung TXT in das Hauptverzeichnis von B:. Hätte man den Backslash hinter B: nicht angegeben, würde in das Verzeichnis auf der Diskette in B: kopiert, das über CD für dieses Laufwerk eingestellt wurde.

Nun ist es an der Zeit, die von uns angelegte Verzeichnis-Struktur wieder zu löschen.

```
CD \TEST\TURBO
Del *.*
CD ..
RD TURBO
Del *.*
CD ..
RD TEST
```

Immer wenn Sie Del *.* aufrufen, werden Sie zur Sicherheit nochmals gefragt, ob Sie denn auch wirklich alles löschen wollen.

In MS-DOS spielen ASCII- und Batch-Dateien eine große Rolle. In diesem Zusammenhang ist der TYPE-Befehl sehr interessant, der wie im CLI eine Textdatei ausgibt. Hex-Dumps sind allerdings nicht möglich.

Möchte man eine kleine ASCII-Datei anlegen, muß man dazu nicht extra einen Editor laden. Man kann auch, ähnlich wie beim Amiga, den COPY-Befehl für diesen Zweck mißbrauchen. Anhand eines Beispiels werden wir Ihnen das nun vorführen:

```
COPY CON DATEI.TXT
```

Alles, was Sie nun über die Tastatur (CONsole) eingeben, wird in das File DATEI geschrieben, das mit der Kennung TXT versehen wird.

Möchten Sie die Eingabe beenden, müssen Sie Ctrl Z (oder F6) und anschließend Return drücken.

Ähnlich der StartUp-Sequnce beim Amiga gibt es auch ein Testfile beim PC, das nach dem Start des Betriebssystems aufgerufen wird. Sein Name lautet

```
AUTOEXEC.BAT
```

Es kann sein, daß auf Ihrer MS-DOS-Diskette diese Datei fehlt. Spätestens dann, wenn Sie versuchen, sich mit

```
TYPE AUTOEXEC.BAT
```

den Inhalt anzeigen zu lassen, werden Sie das feststellen können.

Möchten Sie sich rasch ein neues AUTOEXEC.BAT schaffen, können Sie den oben vorgestellten COPY-CON-Aufruf verwenden:

```
COPY CON AUTOEXEC.BAT
```

Auch der PC kennt das Problem der verschiedenen Tastaturbelegungen. Sollte es einmal nötig sein, die deutsche Tastatur einzustellen, rufen Sie einfach

```
KEYBGR
```

auf.

Auch MS-DOS kann Daten umleiten. Im Gegensatz zum CLI wird der neue Kanal aber nicht vor, sondern hinter die Parameterliste gesetzt.

```
DIR A:\TEST >LPT1
```

gibt z.B. das Verzeichnis von A:\TEST auf einen Drucker aus. LPT1 (oder auch PRN) vertritt in MS-DOS den Drucker.

Befindet man sich im DOS, kann man die Ausgabe auch mit Ctrl P auf den Drucker umleiten. Ein weiteres Mal Ctrl P und die Bildschirmausgabe läuft wieder normal.

Die Ctrl-Taste spielt beim PC übrigens eine große Rolle. Mit Ctrl C kann man Befehle und Batch-Dateien abbrechen. Ctrl S stoppt die Bildschirmausgabe, bis eine beliebige Taste gedrückt wird.

8. Eine Geschichte über und zum Amiga

Zum Commodore Amiga gibt es mehr zu sagen, als daß es in einem Buch niederzuschreiben wäre. Dieses Buch versucht, ein breites Spektrum der Anwendung, Hard- und Software in seinen Kapiteln abzudecken. Als Abschluß möchten wir handfeste Informationen zum Amiga 500 bieten, die endlich einmal einen großen Gesamtzusammenhang bringen und nicht wie sonst nur fragmentarisch einzelne Ausschnitte darstellen.

Dieser vierte Teil des "Großen Amiga 500 Buches" erklärt in kleinen Schritten die Konzeption des Amiga: Seinen Hardware-Aufbau, die Idee, die hinter der Benutzer-Software steht, und schließlich das Betriebssystem und seinen Aufbau. Gerade dieser letzte Teil ist besonders interessant für diejenigen, die schon ganz gespannt darauf sind, zu erfahren, wie der Amiga eigentlich intern organisiert ist. Dabei wollen wir nicht mit Programmbeispielen aufwarten, sondern mit Beispielen aus dem täglichen Leben erklären, warum und wieso das Betriebssystem gerade so und nicht anders arbeitet.

Die Vorgeschichte

Der Amiga, sein Aufbau und seine Anwendungen wurden schon in vielen Büchern beschrieben. Was bisher immer fehlte, war eine übergreifende Beschreibung seines Aufbaus. Welchem Konzept unterliegt die Hardware und welchem die Software? Wie greifen die einzelnen Module des Betriebssystems ineinander und wie hat man es geschafft, daß es fast beliebig erweitert werden kann? Dieses Kapitel gibt Aufschluß darüber, was sich die Fachleute beim Commodore Amiga dabei gedacht haben und wie der Amiga organisiert ist.

Es war einmal vor gar nicht langer Zeit, da wurde ein Computer der Renner auf der ganzen Welt. Das neue Modewort Home-Computer tat sich auf, und jederman hatte einen. Er konnte 16 Farben mit 40 mal 25 Zeichen darstellen. Es gab auch einen Grafikmodus. Doch nach und nach war dieser Computer immer mehr ausgereizt. Die Rede ist vom guten, alten Commodore 64. Mit

ihm hat die Commodore-Ära begonnen, und sie soll weitergeführt werden mit einem Super-Computer, dem Amiga!

Der Amiga ist aus der Idee heraus entstanden, einen preiswerten Computer herzustellen, der eine brillante Farbgrafik und bestechend klaren Sound liefert, einfach schnell ist und sich ausgezeichnet auch von Anfängern bedienen läßt. Dafür setzte sich eine Crew zusammen, die all ihr Wissen in einen Topf warf. Herausgekommen ist damals der Amiga 1000, der mit seinem ladbaren Betriebssystem eine Weltneuheit auf dem Computermarkt in dieser Leistungs- und Preisklasse darstellte. Doch gehen wir Schritt für Schritt vor!

Zuerst wollen wir uns die Hardware des Amiga ansehen, sein Äußeres und sichtbares Inneres und überlegen, wie wir es erweitern können.

8.1 Was man in den Fingern hält: Die Hardware

Der Amiga ist von seiner eigentlichen Konzeption aufgebaut wie ein Mensch. Er besitzt in seiner Grundaustattung, von der wir zuerst sprechen wollen, Sinnesorgane, die ihm Informationen liefern wie z.B. die Tastatur oder die Maus, und er hat auch Möglichkeiten, sich verständlich zu machen über den Monitor, die Lautsprecher oder einen zusätzlichen Drucker.

Auch genau wie ein Mensch benötigt der Amiga Nahrung zum Leben. Dies liefert ihm der Trafo (dies ist die schon fast liebevolle Abkürzung für Transformator), durch den er seinen Strom in der richtigen Form bekommt. Das, was bei uns die Blutbahnen besorgen oder die Nerven, erledigen bei einem Computer die Kabel. Sie übergeben die Nährstoffe, den Strom, oder leiten die Informationen, die Impulse, weiter.

Die Tastatur

Zur Vermittlung von Texten und Informationen ist sicherlich die Tastatur das wichtigste Medium. Auf ihr können wir mehr oder weniger schnell Buchstaben und Zeichen eingeben. Dazu stehen uns 96 mechanische Hub-Tasten zur Verfügung. Aber auch das war nicht immer so. Ältere Computer hatten anstatt mechanischer Tasten, die man richtig herunterdrückt, Folientasten, bei denen

durch den Druck auf eine Fläche zwei Folien verbunden wurden, auf denen elektrische Leiter aufgebracht waren.

Die moderne Amiga-Tastatur ist anders. Sie ist nach DIN aufgebaut. Das heißt, wir haben es mit einer genormten Tastatur zu tun, die dementsprechend komfortabel ist. Jeder, der es gelernt hat, Schreibmaschine zu schreiben, kann ohne große Umstellung auf ihr arbeiten, da jede elektrische Schreibmaschinentastatur nicht anders konzipiert ist.

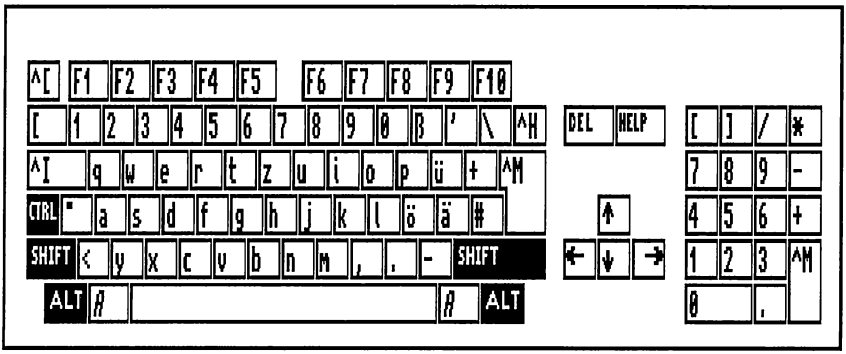


Abb. 8.1: Die 500er-Tastatur

Zusätzlich zu den normalen Tasten, die für das Alphabet und die Zahlen zuständig sind, gibt es noch einige Sondertasten. Der Wagenrücklauf ist durch die [Return]-Taste ersetzt, und das Umschalten auf Großschrift wird mit den beiden [Shift]-Tasten bewerkstelligt. Neu sind z.B. die [Alt]- oder die [Amiga]-Tasten, die aber ähnliche Funktionen wie [Shift] haben, nur daß man hier wieder in ganz andere Ebenen der Tastatur kommt.

Das möchte ich kurz erklären: Jede Taste hat, wenn man sie alleine betätigt, ein ganz bestimmtes Zeichen zugewiesen bekommen, das sie dann weiter an das laufende Programm sendet. Drücke ich nun gleichzeitig dazu die [Shift]-Taste, so schalten alle betroffenen Tasten auf andere Zeichen um. Das bedeutet für uns ganz oberflächlich gesehen, daß wir nun "große" Buchstaben erhalten. Nichts anderes machen die Tasten [Alt], [Amiga] und [Ctrl]. Allerdings wird hier nicht auf Großschrift umgestellt. Bei [Alt] bekommen wir einerseits die zugelassenen Sonderzeichen des Zeichensatzes oder aber die amerikanische Belegung der Tastatur. Dies erfolgt alles ohne Rücksicht auf das, was auf den Tasten steht. Mit den [Amiga]-Tasten löst man hauptsächlich Be-

fehle auf, weil diese extra dafür vorgesehen sind. Das ist aber von Anwendung zu Anwendung verschieden. Auch die Kombination mit [Ctrl] sendet Kommandos, diese wiederum stehen aber fest und können so als Standard verwendet werden.

Bei dieser Besprechung der Umschaltung der Tastaturbelegung sind wir ganz von den Sondertasten abgekommen. So haben wir beim Amiga zusätzlich noch die sogenannten Funktionstasten. Diese liegen oberhalb der Zahlenreihe und sind von 1 bis 10 durchnumeriert. Wie es der Name schon sagt, sind die Tasten für Funktionen vorgesehen, die man über einen einfachen Tastendruck abrufen kann. Aber auch hier ist keine feste Belegung vorgegeben, sondern das jeweilige Programm, mit dem man gerade arbeitet, ist für die Abfrage verantwortlich. Manche Textverarbeitungsprogramme legen auf jede Funktions-Taste einen Text, damit man sich die Tipp-Arbeit ersparen kann, andere lassen so Funktionen des Programms aufrufen. Beide Methoden sind häufig vertreten. Nutzen Sie dies, es spart viel Zeit!

Etwas weiter rechts auf der Tastatur finden wir Tasten, die bei einer normalen Schreibmaschine nicht zu finden sind. Da haben wir zuerst die Cursortasten, mit denen man - es ist kaum zu glauben - den Cursor bewegen kann. Die Richtung ist deutlich zu erkennen. Darüber sind wieder zwei Sondertasten. Die eine, [Del], ist nur das Gegenstück zur Lösch-Taste [Backspace]. Die eine nimmt das Zeichen rechts vom Cursor, die andere das links bzw. an Cursorposition weg.

Sehr nützlich ist auch der Ziffernblock ganz rechts. Er erleichtert die Arbeit besonders bei der Eingabe von vielen Zahlen. Gerade Kalkulationsprogramme oder andere, die viele Zahleneingaben fordern, sind durch den Ziffernblock viel einfacher zu bedienen. Deshalb sind dazu noch die wichtigsten Rechenzeichen der Grundrechenarten angeordnet. Außerdem gibt es die beiden Klammer-Typen rund und eckig sowie die [Enter]-Taste, die gleichbedeutend zur [Return]-Taste ist. Sie wird mit [Enter] bezeichnet, weil es sich um Zahlen handelt. [Return] ist mehr für die Texteingabe gedacht.

Damit haben wir die Tastatur unseres Amiga 500 abgeleuchtet. Ich hoffe, daß Ihnen jetzt einige Zusammenhänge klar geworden sind, denn wir sind die neuen Tasten in der Reihenfolge durchgegangen, in der sie auch zusätzlich dem Computer gegeben wurden. Wir befinden uns momentan am Ende einer Entwicklung, die nicht immer so komfortabel aussah. Angefangen hat alles mit mechanischen Tasten, die ähnlich wie die einer heute alten mechanischen

Schreibmaschine aussahen. Dann wurde mit der Zeit die Mechanik durch Elektronik ersetzt. Aber hier gab es auch Stiefkinder, wie z.B. die Folientasten, die - weil sie keinen Druckpunkt hatten - es dem Benutzer sehr schwer machten. Und je größer die Anforderungen wurden, desto mehr Tasten mußten auch untergebracht werden. Die Tastatur wurde immer umfangreicher, was es nicht zuletzt erforderlich machte, daß man Kurse zur Bedienung einer Computer-Tastatur belegen sollte. Es wurden aber auch viele positive Schritte gemacht! Man hat gelernt, die Tastatur auf ihre Anwendung auszurichten (siehe dazu den Ziffernblock) und sie so anwenderfreundlich zu gestalten. Na dann ...

Die Maus

Als zusätzliches Eingabegerät bietet der Amiga die Maus. Sie stellt im Gegensatz zur Tastatur eine intuitive Eingabe dar. Haben wir über die Tasten konkrete Texte eingegeben, so bewegen wir die Maus über unsere rechte Hand (bei Linkshändern ist es natürlich die linke) und synchron dazu einen Pfeil auf dem Bildschirm. Wie von "Geisterhand" können wir so mit dem Pfeil Objekte auf dem Bildschirm anfahren. Dieses System ist keine Idee der Amiga-Entwickler. Es wurde vom Apple Macintosh kopiert, der als erster Computer mit einer solchen Bedieneroberfläche ausgestattet wurde.

Verschiedene Symbole auf dem Bildschirm stellen die vom Benutzer auswählbaren Funktionen dar. Durch Bewegen des Mauspfeils auf ein Objekt gibt man zu erkennen, daß man diese Funktion aufrufen möchte.

Das alleine reicht zur Informationsübermittlung noch nicht aus, deshalb besitzt die Maus noch zwei Knöpfe. Einen linken und einen rechten, jeder mit anderer Bedeutung. Der linke Mausknopf aktiviert das Objekt, das sich unter der Spitze des Mauspfeils befindet. Man kann so z.B. eine Diskette auf der Workbench oder ein Programm anwählen. Genauso können wir natürlich auch Gadgets (das sind die kleinen Klickfelder in Windows) anwählen, die dann wiederum Funktionen aufrufen. Sie sehen, was man mit einem kleinen Klick alles anstellen kann. Gleichermaßen schwergewichtig in den Auswirkungen ist die rechte Maustaste. Sie läßt, wenn man sie gedrückt hält, die oberste Bildschirmzeile verschwinden und ersetzt diese durch die sogenannte Menüleiste. In ihr finden wir einige Begriffe, denen Funktionen der Workbench oder eines Programms zugeordnet sind. Fahren wir ein Menü mit der Maus ab

und lassen die rechte Taste auf einem Menüpunkt los, haben wir ein Menü ausgewählt. Das ist - hat man sich erst einmal mit der Methode vertraut gemacht - viel einfacher als das Eingeben über die Tastatur, weil man sich bei jedem Tastendruck vertippen kann, nicht so bei der Maus, da hier nur aus vorhandenen Funktionen ausgewählt wird.

Die Maus selbst besteht aus einer schweren Kugel, die im Gehäuse festgehalten wird. Bewegt man die Maus, bewegt sich auch die Kugel und übermittelt über zwei Rollen im Innern die Information der Bewegung. An diesen Rollen sind Scheiben befestigt, die ein gleichmäßiges Lochmuster am Rand tragen. Durch diese Löcher kann Licht von Lichtschranken nur impulsweise fallen. Dadurch entsteht an der Maus-Datenleitung eine Impulsfolge, die genau der Bewegung der Maus entspricht. Über die Geschwindigkeit der Impulse kann nun der Amiga schließen, wie schnell die Maus bewegt wurde und steuert so den Mauspfeil.

Der Trafo

Die Stromversorgung ist beim Amiga 500 separat gelöst. Während beim Amiga 1000, dem Grundmodell, noch alles in einem Gehäuse steckte, ist dies aufgespalten worden. Das hat mehrere Gründe:

Zum einen erspart dies eine Lüftung, denn die Wärmeentwicklung bei der Stromumsetzung ist nicht gering. (Das ist übrigens auch der Grund, warum man den Trafo auf keinen Fall auf empfindlichen Teppichboden stellen sollte!) Es entfällt der Preis für den Ventilator. Der zweite Grund war der fehlende Platz. Entweder hätte man das Diskettenlaufwerk entfernt oder das Gehäuse vergrößert. Einer Vergrößerung widerspricht aber jedes Denken über Proportionen. Schließlich ist in der Breite nichts mehr drin, und auch die Höhe ist genau bemessen. Bleibt nur noch die Tiefe, die aber aus Platzmangel, den bestimmt einige erleiden, auch nicht angegriffen werden kann. Es bleibt wirklich nur noch der externe Kasten, der den Amiga 500 deshalb auch um einiges preiswerter gemacht hat.

Der Amiga benötigt für seinen Betrieb nicht die Spannung, die aus der Steckdose kommt. Deshalb wird diese mit dem Trafo unter Wärmeentwicklung umgesetzt. Das äußerst dicke Kabel rührt daher, daß nicht nur eine Spannung

gefragt ist. Es werden verschiedene gebraucht: der Prozessor benötigt eine andere als die Ein-/Ausgabe-Leitungen usw.

Der Monitor

Als erstes und wichtigstes Ausgabe-Gerät fällt jedem sofort der Monitor ein. Es gibt dafür mehrere Möglichkeiten. Entweder hatten Sie nach dem Kauf des Amiga nicht mehr das Geld, um sich noch einen speziellen Computer-Monitor anzuschaffen und haben deshalb den Fernseh-Modulator gekauft, Sie arbeiten also (wahrscheinlich im 60 Zeichen-Modus) mit einem handelsüblichen Fernsehgerät. Diese Lösung ist sicherlich sehr preiswert und erfüllt zuerst auch ihren Zweck, doch muß an dieser Stelle, mit Rücksicht auf die Augen, darauf hingewiesen werden, daß dies allenfalls bei Spielen akzeptabel ist. Sobald Sie mit Texten in Programmen arbeiten und z.B. beginnen, zu programmieren, sollten Sie an einen richtigen Monitor denken.

Deshalb ist die zweite Möglichkeit ein beliebiger Computer-Monitor, der sich an den Amiga anschließen läßt. Achten Sie hier besonders auf eine richtige Beratung! Je nach Anwendung sollte man nämlich einen monochromen oder mehrfarbigen Monitor wählen. Auch die Frequenz, mit der das Bild dargestellt wird, hängt von der Anwendung ab. Für seltene Arbeiten und vielleicht ein paar Spiele reicht ein billiger Farbmonitor aus. Schreiben Sie aber ihre gesamten Texte mit dem Amiga oder arbeiten Sie allgemein sehr viel mit unserem Lieblings-Computer, dann ist ein gutes Modell, d.h. ein Modell mit einer hohen Auflösung und einer hohen Frequenz, am besten geeignet. Die einfachste Methode dafür ist auch gleich der dritte Weg, der sich beim Monitor-Kauf auftut, nämlich der Commodore-Monitor. Er paßt garantiert zum Amiga und erfüllt für alle mittelstarken Anwendungen seinen Zweck ausgezeichnet.

Die technischen Vorteile zwischen Monitor und Fernsehgerät sind sehr entscheidend. Der Amiga gibt seine Bildinformationen als direkte Anweisungen an den Monitor über das Kabel weiter. Dieser führt sie nur aus und stellt das Bild dar. Bei Fernsehern läuft das ganze Übertragungssystem über einen Umweg. Die Bildinformationen werden erst in eine sendefähige Form übertragen, wie man sie auch über die Dachantenne oder das Kabelnetz empfangen kann. Dann werden sie vom Empfangsteil des Fernsehers wieder in die reinen Darstellungsinformationen umgewandelt und auf der Bildröhre abgebildet. Dieser Umweg ist nicht nur technisch aufwendig, sondern er vermindert leider die

Bildqualität, weil bei jeder Umwandlung einige kleine Störungen und Fehler mit eingebaut werden. Deshalb ist der direkte Weg, wie so oft, der bessere.

Aber nicht gleich jeder Monitor ist ein guter Monitor! Hier hilft nur die Sehprobe!

Die inneren Bestandteile

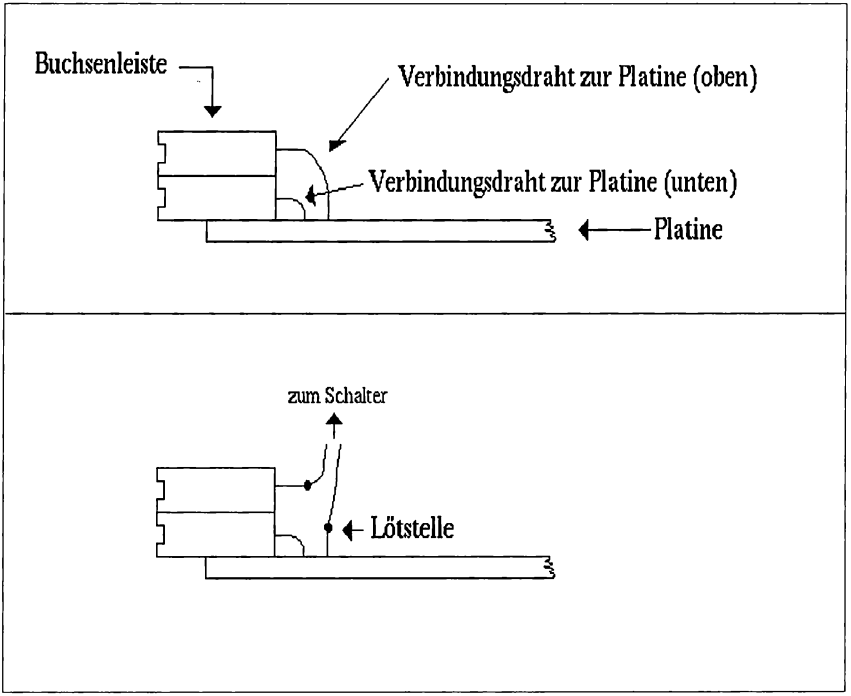


Abb 8.2: Die Platine

Nach diesen Informationen zu den Äußerlichkeiten des Amiga 500 möchte ich Sie jetzt in eine etwas märchenähnliche Welt entführen. In Gedanken werden wir in unseren Amiga einsteigen (am besten gelingt dies über die Maus: Klicken Sie sich über eine Maustaste in die Datenleitung und schon werden Sie zum Ein-Ausgabeport transportiert).

Nachdem wir zuerst die "Sinnesorgane" des Amiga besprochen haben, wenden wir uns jetzt den inneren Organen zu. Da haben wir neben Herz noch Nieren, Milz und auch den Magen, aber natürlich auch noch ein Gehirn. Sehen wir uns zuerst aber das lebensbringende Organ an, das Herz:

Der Motorola 68000-Prozessor

Hat man den Amiga 500 geöffnet (das sollten Sie nicht unbedingt machen, weil dann die Garantie verloren geht. Deshalb haben wir auch ein Photo von der Platine abgedruckt), so sieht man auf der linken Seite (man schaut von vorne auf das Gerät) den längsten und größten aller Chips. Es ist das Herz unseres Amiga, der Prozessor. Er führt alle elementaren Befehle aus und übergibt bestimmte Aufgaben an die Co-Prozessoren. Dadurch ist unser Amiga so schnell. Der eigentliche Prozessor übernimmt nur einen Teil der anfallenden Arbeiten, aber darauf kommen wir später noch.

Der M68000, so die Listenbezeichnung, ist ein Chip der 68000er-Familie, die sich durch hohe Geschwindigkeit auszeichnet. Er arbeitet mit einer Taktfrequenz von 7,16 MHz, die ihm durch den Quarz unterhalb von ihm angegeben wird.

Dieser ist allerdings mit 28,63 MHz getaktet, was für die Videoausgabe gebraucht wird. Auf die Prozessor-Frequenz kommt man erst, wenn man durch 4 teilt. So wird ein Quarz für mehrere Takte benutzt, wie praktisch!

Der Grund, warum der Prozessor so nahe am Expansionport liegt (das ist der Ausgang an der linken Seite des Amiga 500) ist der, daß dort alle wichtigen Prozessor-Leitungen herausgeführt werden.

Um möglichst wenig Verlust dabei zu haben, ging man nach der Devise: je näher, desto besser, denn bei den hier vorliegenden hochfrequenten Strömen gehen bei langen Leitungen die Informationen verloren.

Vorhin wurde gesagt, daß der 68000er Aufgaben an die anderen Prozessoren verteilt. Sehen wir uns jetzt an, welche Spezialaufgaben von den Custom-Chips oder auch Co-Prozessoren erledigt werden.

Die Co-Prozessoren (Custom-Chips)

Schauen wir uns nun die Co-Prozessoren an, die spezielle Arbeiten übernehmen sollen. Als erstes finden wir auf der Platine oberhalb des Grafik-Chips den Co-Prozessor mit Namen Denise. Er ist für sämtliche grafischen Ausgaben zuständig. Das heißt, Denise organisiert die Screens, die Sprites und alle Farbeinstellungen. Zur Grafik gehört aber auch die Verwaltung des Grafik-Speichers, obwohl diese Aufgaben nicht im Bereich von Denise liegen, da sie wirklich nur die Ausgabe bearbeitet. Es gibt dafür einen extra Chip, den Fat Agnus. Damit wird der "fette" Chip rechts neben dem Haupt-Prozessor bezeichnet. Es ist im eigentlichen Sinne nicht ein Chip, sondern drei, die ineinander kombiniert sind. Zum einen der Blitter, der mit unglaublicher Geschwindigkeit Speicherbereiche verschieben oder mit Werten füllen kann, dann der Copper, der für die Beeinflussung der Grafikausgabe zuständig ist, und zum anderen die DMA-Logik, mit der bestimmte Speicherbereiche direkt angesprochen werden können.

Die Kombination von drei Aufgaben in einem Chip hat mehrere Vorteile. Bestimmte Ein- und Ausgänge entfallen, da die Weiterleitung zum anderen Prozessor intern übernommen werden kann. Man kennt das von Telefonverbindungen: Ein Ortsgespräch ist meist klar und deutlich zu führen, aber sobald man versucht, mit weiter entfernten Orten Kontakt aufzunehmen, wird die Verbindung schlechter. Man bräuchte Verstärkerstationen. Dieses Problem wurde beseitigt. Ein weiterer positiver Punkt macht sich im geringen Platzverbrauch bemerkbar und im Geschwindigkeitsvorteil. Also können wir rundum zufrieden sein, daß unser Amiga 500 diesen kompakten Chip besitzt.

Zu unseren beiden Co-Prozessoren gesellt sich noch ein dritter mit Namen Paula. Er betreut sämtliche Ein- und Ausgabeoperationen in Zusammenarbeit mit den beiden Ausgabe-Chips, die wir oberhalb und rechts oben von Paula finden. Paula selbst liegt übrigens direkt über der Fat Agnus. Die Gebiete von Paula sind die Diskettensteuerung, Interrupts, Tonausgabe über die Audio-Ausgänge und Datenübertragung mit dem seriellen Port. Der letzte und vierte Co-Prozessor in der Runde ist Gary. Ein Chip, dessen Namen von seiner Funktion herrührt, denn Gary ist ein Gate-Array-Chip. Das heißt, er ist für aufwendige, logische Funktionen und Operationen da.

Diese vier Chips unterstützen also unseren Amiga 500 Haupt-Prozessor, den 68000er. Wenn Arbeiten anfallen, bei denen Speicherbereiche verschoben oder kopiert werden müssen, wird der Fat Agnus angewiesen, dies zu tun. Der

68000er hat während dieser Zeit anderes im Kopf und erledigt schon die neuen Befehle. Ein anderes Beispiel sind die Diskettenfunktionen, die auch nicht vom Haupt-Prozessor bearbeitet werden. Sofort gibt der 68000er die Befehle an Paula weiter. Hier ist allgemein alles gut aufgehoben, was mit Ein- und Ausgabe zu tun hat. Nur durch diese geschickte Arbeitsteilung kann der Amiga noch im Multitasking-Betrieb Schnelligkeit beweisen!

Der Speicher

Neben einigen anderen kleinen IC fällt besonders eine Gruppe unterhalb des Fat Agnus auf. Zwei Reihen à acht Chips bilden den Grund-Speicher des Amiga 500. Es sind 512 KByte, die in 256-Byte-RAM-Bausteine aufgeteilt sind. Sie haben eine Zugriffszeit von 150 Nanosekunden, was zwar nicht berauschend schnell ist, aber ausreicht. Es gibt noch höhere Geschwindigkeiten, die man bei Speichererweiterungen findet. Üblich sind z.B. 120, 100 oder sogar 80 Nanosekunden. Dieses RAM hat noch eine versteckte Besonderheit, und zwar können nur auf diesen RAM-Bereich die Custom-Chips, also die Co-Prozessoren, zugreifen.

Jede Speichererweiterung bietet nur zusätzlichen Speicher für den Haupt-Prozessor. Das muß man bedenken, wenn der Speicher für Daten verteilt wird. Denn nur hier können Daten für Grafik, Sound oder I/O-Kommunikation untergebracht werden, weil auch die Custom-Chips darauf Zugriff haben müssen. Liegen die Daten aber in einem anderen Speicherbereich, ist es wie mit kleinen Kindern, die an etwas heranwollen, was zu hoch gestellt ist: sie erreichen es nicht und fallen herunter. Das macht auch der Amiga und zwar sehr deutlich nach diesem Bild. Er stürzt ab!

Beim Blick auf die Platine fällt dann natürlich auch der leere Ausschnitt auf der rechten unteren Seite auf. Hier kann die interne Speicher-Erweiterungs-Karte eingesteckt werden, damit man 512 KByte mehr RAM erhält, den, wie oben erwähnt, die Custom-Chips nicht nutzen können.

Dafür ist dieser Speicher im Zugriff auch um einiges schneller. Nicht, weil die Zugriffszeit von 150 Nanosekunden übertroffen wird, sondern weil keine Wait-States für die Custom-Chips verloren gehen. Jeder Zugriffs-State kann vom 68000er - und nur von ihm - ausgenutzt werden.

I/O, die Ein- und Ausgänge

Sinnlos wäre die Arbeit eines Computers, wenn er die Ergebnisse seines Schaffens nicht an den Benutzer weitergeben könnte, und noch sinnloser wäre es, wenn er gar keine Daten für die Berechnungen bekäme. Deshalb finden wir auf der Rückseite zehn Ein- bzw. Ausgänge, an denen Informationen anliegen oder abgefragt werden. Außerdem haben wir auf der linken Seite den Systembus durchgeführt, über den wir Erweiterungen, die sogar den Haupt-Prozessor betreffen, anbauen können.

Sehen wir uns jetzt zuerst die hinteren Stecker an. Als Hilfe sei noch gesagt, daß es sich um einen Stecker handelt, wenn die Kontakte als Stifte frei liegen. Als Buchse bezeichnet man einen Ein- bzw. Ausgang, bei dem die Kontakte nicht frei liegen, sondern vertieft sind (einfachstes Beispiel dazu sind die Steckdosen [Buchsen] für die 220V-Stromversorgung und die dazugehörigen Stecker in jedem Haushalt).

Betrachtet man den Amiga 500 von hinten, so sieht man links zwei ziemlich gleich aussehende, eingebaute Stecker. Sie sind für die Maus oder den Joystick gedacht. Beide Eingabegeräte sollten Ihnen bekannt sein. Die Maus wird für die Bedienung des Amiga lebensnotwendig gebraucht, und der Joystick hilft einem bei Spielen, seinen Mann zu stehen. An dieser Stelle nimmt der Amiga also Informationen ab. Es handelt sich um einen Eingang. Nebenbei liegen dort auch noch 5 Volt an, die als Stromversorgung genutzt werden können (man beachte, daß es sich um 125 mA handelt!).

Weiter rechts daneben sind wiederum zwei gleiche Buchsen, die hier allerdings nicht als Eingang, sondern nur als Ausgänge fungieren. Hier können über Cinch-Stecker Stereo-Anlage, Monitor oder anderes Gerät angeschlossen werden, um die Töne, die der Amiga produziert, abzunehmen. Schließen Sie den Amiga-Monitor an, so finden Sie am Scart-Kabel die dazu passenden Stecker. Möchten Sie aber an ihre Stereo-Anlage gehen, dann müssen Sie sich nur vier Stecker und ein genügend langes Audio-Kabel kaufen. Mit etwas Lötarbeit oder einem guten Freund ist die Leitung schnell gefertigt, und der Amiga-Sound ist im Bereich von bis zu mehreren 100 Watt zu hören! Der Vorteil liegt dabei auf der Hand! Bessere Klangqualität durch hochwertige Boxen im Vergleich zu den Mini-Lautsprechern des Monitors, ein größeres Volumen und eine höhere Klangbrillanz, weil Sie alle Funktionen ihres Verstärkers nutzen können.

Der erste etwas kompliziertere Stecker wird für die nächste Buchse gebraucht. Wir haben es mit 23 Polen zu tun, die hier aus dem Amiga geführt werden. Aber wozu? Ja, damit ist es ermöglicht worden, bis zu drei weitere Disketten-Laufwerke anzuschließen. Hier wird also eine Verbindung zu Paula, dem Ein- und Ausgabe-Chip, hergestellt. Neben den Datenleitungen finden wir hier zwei Pins, die mit 5 bzw. 12 Volt belegt sind, und mehrere Masse-Pole. Beachtenswert ist die Einrichtung, den Stecker, wenn er denn angesteckt wird, mit zwei Schrauben zu befestigen, denn im allgemeinen lösen diese sich sehr schnell, und damit ist die Gefahr gebannt.

Noch ein Ein- bzw. Ausgang von ähnlichem Kaliber ist der rechte davon. Wir haben die serielle Schnittstelle vor uns. Mit ihr werden Daten aus dem Amiga heraus und in ihn hinein getragen. Die Bezeichnung "seriell" stammt von der Art, mit der die Daten ein- und ausgeführt werden. Sie werden wie in einer Serie hintereinander versandt. In diesem Fall haben wir wieder einen Stecker vor uns. Jeder seiner 25 Pins hat eine genormte Belegung. Diese Norm ist mit dem Namen RS-232 versehen, damit jeder weiß, welche Aus- und Eingänge an den Leitungen zu finden sind. Obwohl diese Voraussetzungen einen reibungslosen Betrieb erlauben, wird die Datenübertragung hierdurch nur ungern gewählt. Das Übertragungsverfahren ist nämlich nicht besonders schnell. Sie können sich vorstellen, daß es effektiver wäre, mehrere Daten gleichzeitig zu senden und zu empfangen.

Und damit wären wir bei dem nächsten Ausgang. Im Gegensatz zur seriellen Schnittstelle ist dies hier die parallele. Sie hat die gleiche Pin-Anzahl, und damit man die beiden nicht verwechselt, wurde dieser als Buchse konstruiert. Es liegen neben einer Spannung, der Erde, noch einige Statusleitung an diesem Anschluß. Das wichtigste von allem sind aber die 8 Datenleitungen. Wir haben damit eine achtfache Übertragungsrate, was sich auch wirklich bemerkbar macht. Schließt man seinen Drucker einmal an die serielle und dann an die parallele Schnittstelle an, wird man sofort erkennen, wo die Datenübertragung schneller abgewickelt wird. Auch hier haben wir es wieder mit einem Standard zu tun. Es ist der berühmte Centronics-Standard, der auch von allen IBM-kompatiblen Druckern und Computern benutzt wird. Ein Drucker oder ein beliebiges anderes Gerät, das man an dieser Schnittstelle betreibt, läßt sich fast überall anschließen.

Der viereckige Ausgang daneben ist sozusagen lebenswichtig für unseren Amiga 500. Ohne die Stromversorgung, die hier angeschlossen wird, läuft nichts. Mit einer Kerbe ist gesichert, daß man den Stecker nicht verkehrt

herum einsteckt. Wie schon oben erwähnt, werden hier nicht nur zwei Leitungen eingeführt, die eine gewünschte Spannung enthalten. Der Amiga benötigt mehrere verschiedene Spannungen, die alle im Netzteil erzeugt werden. Deshalb haben wir hier mehr Kontakte.

Wenn Sie in dem glücklichen Zustand sind, daß Sie einen Monitor besitzen, ist der nächste Ausgang der richtige für Sie. Es handelt sich um den Monitor-Anschluß, an dem ein Videobild mit RGB-Informationen in analogem und digitalem Format anliegt. Das Kabel, mit dem Sie an diesen Ausgang gehen, besitzt auf der einen Seite die Buchse, die am Amiga festgeschraubt werden kann, und auf der anderen einen Scart-Stecker, der leider nicht gesichert ist und deswegen häufig bei kleineren Bewegungen herausrutscht. Achten Sie auch wegen der hohen mechanischen Spannung, die auf diesem Kabel ruht, darauf, daß es möglichst entspannt verwendet wird.

Der letzte Ausgang gibt noch einmal das Video-Signal aus. Hier allerdings in einer anderen Norm. Es handelt sich um die amerikanischen Cinch-Stecker, mit denen das Bild abgenommen werden kann. Der zweite wichtige Unterschied, der zwischen den beiden Video-Ausgängen besteht, ist, daß beim letzteren auch der Ton mit übertragen wird, was bei Scart nicht der Fall ist.

Damit haben wir alle Ein- und Ausgänge besprochen, die man auf der Rückseite des Amiga 500 finden kann. Es gibt aber noch eine weitere Schnittstelle, und zwar den Prozessor-Ausgang an der linken Seite des Amiga. Auch hier liegen sehr interessante Informationen an. Externe Speichererweiterungen und System-Ergänzungen, die direkten Zugriff auf den Prozessor benötigen, sind an diesem Ein-/Ausgang richtig. Durch die direkte Verbindung läßt sich hier alles ergänzen, was den Amiga 500 erweitert oder verändern soll.

Hardware-Erweiterungen

So, wie man den Amiga 500 in der Grundausstattung bekommt, ist er natürlich nicht unbedingt vollkommen. Er enthält alles Nötige, um voll zu funktionieren, und dies auch nicht gerade unkomfortabel, wie Sie im vorhergehenden Text gelesen haben. Aber es gibt noch vieles, was man ergänzen könnte (wie überall).

Deshalb stellt Commodore viele Hardware-Erweiterungen zur Verfügung, die das Leben einfacher, bequemer oder einfach individueller machen. So kann man mit einer zusätzlichen Floppy bestimmt mit einigen Diskettenwechseln weniger leben. Aber auch andere Firmen haben das Geschäft entdeckt und bieten zusätzliche Geräte an, die z.B. aus dem kleinen Amiga 500 einen Speicherriesen machen.

Massenspeicher

Das eingebaute Diskettenlaufwerk - wir nennen es Floppy - ist ein Massenspeicher. Mit diesem Fachbegriff kennzeichnet man alle Geräte, die in der Lage sind, Daten zu speichern. Allerdings differenziert man hier etwas. Denn es gibt auch ganz primitive Speicher wie z.B. eine Glühlampe, die ihren Leuchtzustand speichert, die wir damit nicht meinen. Es müssen größere Datenmengen speicherbar sein, eben Massen von Daten.

Das fest installierte Diskettenlaufwerk ist ein Massenspeicher. Dieser mag zwar zu Anfang ausreichen, aber bei häufigeren Arbeiten und mit höheren Ansprüchen wächst auch der Bedarf an weiteren Floppies. Denn mit nur einem Laufwerk muß man gerade bei der Arbeit im CLI doch sehr häufig die Diskette wechseln und schon die Textverarbeitungen oder Programmiersprachen erfordern eine Datendiskette. Also faßt man den Entschluß, sich ein weiteres Diskettenlaufwerk zuzulegen. Dieses wird über den weiter oben beschriebenen Port an den Amiga 500 angeschlossen und ist dann betriebsbereit. Es erhält seinen Strom über das gleiche (etwas dickere) Kabel, über das auch die Daten ausgetauscht werden.

Als Modelle bieten sich mehrere Varianten an. Zum einen gibt es von Commodore ein 3½-Zoll-Diskettenlaufwerk. Es ist im Amiga-Design gehalten, kostet aber entsprechend etwas mehr Geld. Dann gibt es auch noch Laufwerke von Fremdherstellern, die teilweise qualitativ nicht sehr hochwertig sind, andere aber benutzen Qualitätslaufwerke, bei denen man sicher sein kann, daß die Datensicherheit gewährt ist. Man sollte beim Kauf einfach darauf achten, daß es sich z.B. um ein Markenchassis wie NEC handelt. Hier sind die Modelle 1036 und 1037a weit verbreitet und wesentlich leiser als die Commodore-Floppies. Besonders interessant sind die zusätzlichen Funktionen, die von diesen Diskettenlaufwerken angeboten werden. So gibt es welche, die anzeigen, auf welcher Seite und Spur gerade gelesen oder geschrieben wird, und

die meisten können auch ausgeschaltet werden. Das ist wichtig für Programme, die den Speicher brauchen, der von jedem zusätzlichen Laufwerk verbraucht wird. Man sollte auch auf jeden Fall auf einen durchgeführten Bus achten. So bezeichnet man Hardware, die den Ausgang, an den sie angeschlossen wird, wieder als Ausgang anbietet. Man kann dann weitere Diskettenlaufwerke anschließen.

Neben 3½-Zoll-Floppies werden aber auch 5¼-Zoll-Laufwerke angeboten. Obwohl dieses Format beim Amiga gar nicht üblich ist, wird es trotzdem verwendet. Das liegt zum einen an der Möglichkeit, ein MS-DOS-System zu simulieren. Dann findet man nämlich alle Daten nur noch auf solchen Disketten wieder, die natürlich nicht vom eingebauten Laufwerk gelesen werden können. Ein weiterer Punkt ist der Preisunterschied. Die harten kleinen Disketten sind immer noch mindestens doppelt, wenn nicht dreifach so teuer wie die großen flexiblen Brüder. Das ist ein wichtiger Aspekt, wenn Sie z.B. sehr oft von Datendisketten Sicherheitskopien anfertigen wollen.

Ein ganz anderes Gebiet der Massenspeicher sind die Festplatten. Im Gegensatz zur Diskettenstation kann hier die Diskette, also das Speichermedium, nicht entnommen werden. Festplatten sind fest versiegelte Geräte, die man nicht öffnen kann und sollte. Der zweite Unterschied der Festplatte zum Diskettenlaufwerk liegt einfach in der Speicherkapazität. Auf eine normale Amiga-Diskette passen etwa 880 KByte an Daten. Die kleinste Festplatte faßt aber schon eine Datenmenge von etwa 20 MByte, was mehr als das 23fache der Diskettenkapazität ist.

Leider gibt es aber noch den dritten und entscheidenden Unterschied, den Preis. Ein Diskettenlaufwerk kostet für den Amiga 500 DM 300,-. Die Diskette kostet etwa DM 2,-. Das macht zusammen nicht viel mehr als DM 302,-. Eine äußerst billige Festplatte kostet aber schon mindestens DM 1000,- (die Preise gehen auf und ab!). Wir haben ein Preisverhältnis pro KByte von DM 0,34 zu DM 0,05. Es zeigt sich also rein rechnerisch, daß Festplatten wesentlich billiger sind. Das Problem liegt aber in der Finanzierung. Denn die Festplatte muß sofort beim Kauf finanziert werden und liegt sehr hoch im einmaligen Preis. Bei den Disketten verhält es sich anders, wenn man bedenkt, daß das Laufwerk schon mit dem Computer geliefert wird und die Disketten um DM 25,- pro 10 Stück kosten.

Nicht vergessen werden sollte bei der ganzen Diskussion, daß der Datenzugriff, also die Geschwindigkeit, mit der ich an meine Programme oder

Spiele herankomme, sich sehr unterscheidet. Während eine Diskettenstation, bedingt durch die grobe Mechanik und das Aufzeichnungsverfahren, nur eine sehr langsame Übertragungsgeschwindigkeit erreichen kann, haben Festplatten oftmals Übertragungsraten in der Nähe von Speicherverschiebungen, wir wie sie nur von der RAM-Disk her kennen.

Drucker

Die Auswahl eines Druckers ist beim Amiga keine leichte Aufgabe. Wer es einfach haben möchte, kann sich einen Commodore-Drucker zulegen. Hier gibt es einige Modelle zur Auswahl. Aber viele andere Firmen stellen qualitativ gleichwertige, wenn nicht bessere Drucker her, die man zu sehr günstigen Preisen bekommen kann. Es wäre an dieser Stelle ein vergebliches Unterfangen, zu sagen, welcher der Drucker der richtige ist. Deshalb haben wir uns überlegt, ihnen die Druckertypen zu beschreiben. Sie lernen etwas über die verschiedenen Methoden, mit denen man das Gewünschte auf das Papier bringen kann und haben so gleichzeitig eine Entscheidungshilfe, die sich beim späteren Druckerkauf sicherlich auszahlt.

Damit sich der Drucker an den Amiga 500 anschließen läßt, ist es zuerst wichtig, daß er die richtige Schnittstelle hat. Der Drucker und unser Amiga müssen also die gleiche Sprache verstehen, damit sie kommunizieren können. Entweder man geht über die serielle Schnittstelle, die aber sehr langsam ist, oder man benutzt die parallele. Diese ist auch am meisten verbreitet, weil alle IBM-kompatiblen Geräte die gleiche Centronics-Schnittstelle haben. Das macht auch bei der Beschaffung des Verbindungs-Kabels keine Probleme, denn jedes IBM-Druckerkabel funktioniert.

Für die Auswahl des Druckers gilt es nun, drei wichtige Punkte zu beachten. Zuerst ist das das Druck-System. Es gibt Matrix-, Tintenstrahl-, Typenrad- und Laser-Drucker. Jedes Modell hat dabei seine Vor- und Nachteile. Matrix-Drucker sind besonders schnell, weil der Druckkopf aus lauter Nadeln besteht, die in sehr schneller Geschwindigkeit gegen das Farbband hauen. Die Qualität der Druckers differiert hier aber stark. So gibt es billige Druckköpfe mit nur einer sehr geringen Schriftqualität, weil sie nur wenige Nadeln haben. Eine Stufe besser als die 9-Nadel-Drucker sind die mit 24 Nadeln. Allerdings steigt hier auch der Preis. Es werden sogar schon 48-Nadel-Drucker gebaut. Die sind allerdings schon fast so teuer wie Typenrad-Drucker. Ein Nachteil beider

Druck-Systeme ist ihre hohe Lautstärke-Entwicklung. Durch die vielen Nadeln kann ein Kreissägen-ähnlicher Ton entstehen. Deshalb muß man darauf achten, daß der Drucker gut schallisoliert ist. Auch die Typenrad-Drucker machen Lärm. Dieser hört sich allerdings mehr wie das schnelle Schlagen eines Hammers an.

Viel leiser sind die Tintenstrahl-Drucker. Wie man sich leicht vorstellen kann, entwickelt ein Tintenstrahl nur sehr wenig Lärm. Er schnurrt bestenfalls. Auch die Qualität dieser Drucker ist bemerkenswert. Sie erreichen auf jeden Fall ein höheres Maß, als es die Matrix-Drucker tun. Doch das macht sich auch im Preis bemerkbar. Tintenstrahl-Drucker sind wesentlich teurer als die Matrix-Modelle, da sie einerseits weniger gekauft werden und andererseits eine feinere Mechanik benötigen. Aber auch der Betrieb ist kostenintensiver. Während ein Matrix-Drucker nur normales Endlospapier benötigt, brauchen wir für einen Drucker, der mit flüssiger Tinte arbeitet, spezielles Papier, das besonders saugfähig ist und spezielle Tinte, die besonders schnell trocknet. Ein Matrix-Drucker benötigt nur eine Farbbandkassette.

Hat man einen Laser-Drucker, möchte man sich mit diesem Kleinkram gar nicht mehr abgeben. Er arbeitet mit einem ähnlichen Verfahren wie auch die modernen Kopiergeräte. Eine Walze wird elektrisch geladen. An den Stellen, an denen gedruckt werden soll, wird mit einem Laser-Strahl die Ladung angehoben, wodurch wir das spätere Druck-Bild auf die Rolle bekommen. Diese wälzt sich nun durch die Farbkartusche und stößt die Farbe dort ab, wo sie noch geladen ist. An den ungeladenen Stellen bleibt die Farbe kleben. Die Walze überträgt nun die haftende Farbe auf das Blatt Papier, das an ihr vorbeigeführt wird. Nun wird die Farbe auf dem Blatt Papier nur noch mit großer Hitze eingebrannt. Fertig ist der Druck!

Mit diesem technischen Verfahren ist es möglich - vorausgesetzt, es herrschen die richtigen Hardware-Konfigurationen - gestochen scharfe Grafiken und Texte zu drucken, bei denen man keinen Unterschied zwischen Computer-Druck und Maschinen-Druck feststellen kann. Sicherlich liegt der Preis dieser Geräte schon in einer Ebene, die nicht von jedem erreicht werden kann. Denn wer gibt schon mehrere tausend Mark nur für einen Drucker aus? Aber es ist ein schöner Traum, der für den einen oder anderen unter Ihnen vielleicht sogar einmal Wirklichkeit wird.

8.2 Was den Amiga bedienbar macht: Die Workbench und das CLI

Die Workbench

In den Anfangszeiten der Rechenmaschinen gab es den Komfort, den z.B. der Amiga bietet, nicht! Denkt man nur an die Rechenmaschine von Pascal, bei der man über Räder die Zahlenwerte einstellen mußte, so ist ein enormer Fortschritt zu verzeichnen. Aber auch die Computer der früheren Vergangenheit bieten nicht soviel an Funktion und eine so einfache Bedienung, wie sie uns jetzt vorliegt.

Bei anderen Computer-Systemem war es unumgänglich, daß man die Befehle über die Tastatur in einer Sprache eingab. Dabei treten mehrere leicht verständliche Probleme auf: Zum einen beherrschen nur wenige die Bedienung einer Tastatur. Viele Menschen müssen erst lange suchen, bis sie den gewünschten Buchstaben finden. Nur wer viel Übung hat, kann in kurzer Zeit Texte über die Tastatur eingeben. Zum anderen gibt es aber auch bei jeder Sprache Verständigungsschwierigkeiten. Entweder man ist des Englischen nicht mächtig und versteht so kein einziges Wort, oder aber man kann Englisch, nur wollen einem die nötigen Variationen der Befehls-Worte nicht in den Kopf. Warum heißt es auf dem einen Computer Delete und auf dem anderen Erase, warum nennt der nächste es Kill und wieder ein anderer Discard? Wer soll sich durch so einen Wust an Schattierungen noch durchwühlen? Man verliert die Lust am Computern!

Man überlegte sich dann, wie man den häufigsten Fehlern entgehen könnte. Und heraus kam eine Programmbedienung, bei der an den meisten Stellen nur eine Auswahl der möglichen Funktionen zur Verfügung stand. Es konnte dadurch schon keine Fehleingabe mehr geschehen, weil alles vorgegeben ist. Schritt für Schritt gelangte man so zu dem, was man aufrufen wollte. Allerdings war dieses System sehr mühsam zu programmieren und zu bedienen, denn je ausführlicher das gesamte System wurde, desto endloser waren die Auswahltabellen und desto länger der Weg zum gewünschten Ziel. Ein Nachteil, der sich besonders bei Anfängern zeigte, war der ständige Wechsel zwischen dem Blick auf den Bildschirm und auf die Tastatur. Die meiste Zeit ging damit verloren, den Bildschirm zu sichten oder eine Taste auf der Tastatur zu suchen. Deshalb beschloß man eine weitere Verbesserung:

Die Situation hat sich heute geändert. Gerade weil man diese Probleme erkannt hat, die bei den älteren Computer-Systemen aufgetreten sind, wurde neu überlegt, das Alte über den Haufen geworfen oder weiterentwickelt. Herausgekommen ist eine Bedieneroberfläche, wie sie der Amiga besitzt. Nicht mehr die Tastatur ist das wichtigste Eingabe-Medium, sondern die Maus hat sie abgelöst. Nur noch die Position eines kleinen Pfeils, Cursor genannt, und der Klick auf eine von zwei Tasten löst einen Befehl aus. Gerade diese Einfachheit macht es gerade dem Laien möglich, eine so hochentwickelte Maschine zu bedienen. Die Maus kann man fast blind bewegen, ohne den Kontakt mit ihr zu verlieren. Denn ihre Position wird synchron auf dem Bildschirm über den Pfeil dargestellt.

Nehmen wir uns ein Beispiel, das illustriert, welchen Wandel die Bedienung gemacht hat. Betrachtet man nur den Vorgang, der zum Ansehen des Inhalts einer Diskette notwendig ist, so erkennt man den gravierenden Unterschied. Bei alten Systemen wußte man gar nicht, daß überhaupt eine Diskette vorhanden ist, selbst dies mußte nachgeprüft werden. Beim Amiga wird uns das über die Disketten-Icons auf der Workbench klar dargestellt. Interessiert man sich dann für den Inhalt einer Diskette, so klickt man einfach doppelt auf ihr Symbol, und es öffnet sich ein neues Fenster. Ältere Computer boten hier höchstens eine Bildschirmseite, auf der in mehr oder minder großer Geschwindigkeit zeilenweise die Namen der auf der Diskette gespeicherten Files langliefen. Gerade bei langen Files machte dies einige Probleme. Schließlich waren die Files weder sortiert noch einzeln anwählbar. Man mußte eben schnell nachsehen oder das Verzeichnis wieder von Anfang an betrachten. Die Workbench hingegen geht einen viel komfortableren Weg. Sie zeigt in ihrem Fenster, das separat für das Hauptverzeichnis geöffnet wurde, nur Symbole für die wichtigen Files des Verzeichnisses. Sind auch andere enthalten, die den Benutzer nur verwirren würden, bekommt er diese gar nicht zu sehen, weil für sie kein Bild existiert. Selbst bei einer großen Menge an Symbolen versagt die Workbench nicht. Hier wird einfach das Fenster als Ausschnitt betrachtet, und dieser wiederum läßt sich beliebig verschieben. Sie können in Ruhe alles betrachten und erst dann auswählen.

Drumherum ranken sich die komfortablen Bedieneigenschaften der Benutzeroberfläche: Gadgets zur schnellen Funktionsbearbeitung, Menüs zur Auswahl wichtiger Funktionen und die deutlichen Symbole, die ein außersprachliches Verständnis ermöglichen.

Besonders einfach und gut gelungen ist der Start eines Programms. Hierbei müssen wir lediglich dafür sorgen, daß, wie man sich denken kann, der Inhalt des Verzeichnisses vor uns liegt, in dem das Programm zu finden ist. Wir haben also das Programm-Symbol vor uns. Nun reicht es, das Symbol mit dem Mauspfeil anzufahren und zweimal kurz auf die linke Maustaste zu klicken. Mit diesem Doppelklick haben wir der Workbench mitgeteilt, daß wir das Programm starten möchten. Einfacher geht es wohl kaum. Durch die Einführung des Doppelklicks erspart man sich einige Arbeit. Heute auch noch möglich, aber fast völlig ungenutzt, ist der alte Weg. Bei ihm wurde das Programm-Symbol einmal angeklickt, um es zu aktivieren. Dann ging man auf die Menüleiste und wählte dort den Punkt Open an, woraufhin das Programm geöffnet, also gestartet wurde. Aber Sie sehen selbst, es ist viel zu umständlich.

Das CLI

Sicherlich ist die Workbench gerade für den Einsteiger eine besonders gute Einrichtung. Und auch der Profi möchte sie nicht missen, wenn es um einfache Operationen geht, die sich ohne Mühe ausführen lassen. Stellt man aber hohe Ansprüche oder möchte man in die speziellen Einstellungen gehen, so ist dies einfach nicht mehr mit Objekt-Steuerung zu bewerkstelligen. Der Amiga bietet als Ausweichmöglichkeit die Eingabe über den altbekannten Zeileneditor, wie er von den alten Systemen bekannt ist. Dieser ist natürlich verbessert worden und den Anforderungen eines professionellen Programmierers angepaßt. Wir haben das CLI!

Im CLI-Zeileneditor gibt es eine einfache, aber wirkungsvolle Regel. Man schreibt zuerst das Befehlswort in die Zeile und dahinter die Parameter. Das Befehlswort ist in jedem Fall ein CLI-Befehl oder ein Programm auf der Diskette. Das macht es besonders einfach, wenn man ein Programm starten möchte. Es genügt, den Programmnamen einzugeben und mit [Return] zu bestätigen. Schon wird das Programm geladen und gestartet.

Aber auch Befehle sind sehr einfach zu handhaben. Es gibt Befehle, die (ähnlich wie Programme) ohne jeden Parameter gestartet werden können. Andere wiederum benötigen aber einige Werte, um arbeiten zu können. Für denjenigen, der leicht vergißt, was für Parameter ein Befehl benötigt, gibt es eine besondere Einrichtung im CLI, die Hilfsfunktion. Sie gestattet es, hinter dem

Befehlsnamen ein Fragezeichen anzugeben. Dann erhält man eine Liste der Wertemöglichkeiten oder einen Hilfstext, der beschreibt, was dieser Befehl macht.

8.3 Was den Amiga steuert: Das Betriebssystem

Jeder Computer muß irgendwie gesteuert werden. Viele Benutzer machen sich darüber keine Gedanken, aber alles das, was nach dem Einschalten eingestellt, eingerichtet und initialisiert wird, muß schon von einem Programm gesteuert sein. Und auch die Grafik, das Einschaltbild des Amiga, erfordert großen Aufwand an Programmierung.

Für alle diese Grundaufgaben, für die Grafikausgabe, für das Lesen von Diskette, für den Datenaustausch an sich ist ein Betriebssysten geschaffen worden. Es erledigt die täglichen Aufgaben mit links und verhilft so dem Programmierer dazu, sich mit den wesentlichen Problemen auseinanderzusetzen. Stellen Sie sich vor, ein Programmierer müßte jedes Mal wieder von neuem die Ausgabe programmieren. Das Projekt würde schon im Kern scheitern, weil es einfach zu umständlich ist, immer wieder bei Adam und Eva anzufangen.

Die Vorgeschichte

Am Anfang war der Amiga sehr offen konzipiert worden. Wenn Sie sich noch an das Erstlingsmodell erinnern, wissen Sie, daß dieser Computer nach dem Einschalten wirklich nichts wußte. Er forderte nur mit einem auf ROM gespeicherten Bild dazu auf, die Kickstart-Diskette einzulegen. Auf ihr befand sich zu der Zeit das gesamte Betriebssystem. Man hatte dadurch zwar den Nachteil in Kauf zu nehmen, daß nach dem Einschalten einige Minuten zum Booten des Betriebssystems verloren gingen, aber durch diesen Kunstgriff war es möglich, jederzeit Veränderungen einzubringen. Diese mußten nur auf die Diskette gebracht werden und konnten sofort auch von Ihnen genutzt werden.

Dieses System hatte sich vorzüglich bewährt, doch leider ist nicht nur dies ein Gesichtspunkt, der bei der Entwicklung von Computern bedacht wird. Ein anderer und sehr wichtiger ist der Preis eines Computers. Dieser konnte aber

noch mehr verringert werden, wenn man eine einigermaßen gute Version des Betriebssystems auf ein ROM brennt und dafür die Ladevorrichtung entfernt. Es waren der Amiga 2000 und auch Ihr Amiga 500 geboren. Beide haben das Kickstart, das ein Amiga-1000-User auf der Diskette hat, im ROM. Und zwar ist dies die Version 33.180, wie Sie ganz leicht über den Workbench-Menüpunkt Extras-Version abrufen können.

Übrig geblieben ist der Vorgang, bei dem die Workbench gebootet wird. Beim Booten werden die Befehle ausgeführt, die in der Startup-Sequence stehen. Dadurch läßt sich der Amiga je nach Boot-Diskette persönlich einrichten. Auf der einen Diskette wird mittels der Startup-Sequence ein System eingerichtet, das für die Arbeit im CLI gedacht ist, auf einer anderen startet diese Sequence andere Programme, wodurch einige Arbeit gespart wird. Diese Möglichkeit macht es besonders für Benutzer einfach, die nicht so viel von ihrem Computer verstehen. Und der Programmierer sieht hier einen Weg, sich Arbeit abzunehmen.

Uns soll jetzt interessieren, wie unser Betriebssystem aufgebaut ist. Dazu werden wir uns erst einmal ansehen, was es alles kann. Dabei beginnen wir mit dem Start-Vorgang:

Nach dem Einschalten des 500er erscheint die bekannte Grafik mit der Workbench-Diskette, die eingelegt werden soll. Da das Betriebssystem diese Grafik ausgeben kann, läßt dies erkennen, daß schon die Grafik-Routinen vorhanden sind. Sie sind im Kickstart-ROM untergebracht. Genauso läßt sich auch feststellen, daß die Disketten-Operationen vom Kickstart gesteuert werden. Woher könnte der Amiga sonst wissen, wie er den Boot-Block von der Diskette lesen sollte? Auf die gleiche Art kann man dann schließlich noch erwähnen, daß auch Intuition, die Amiga-Benutzeroberfläche, in das ROM gebrannt wurde, denn sofort, nachdem der Boot-Block geladen wurde, erscheint die Workbench-Screen mit dem AmigaDOS-Window. In dem Boot-Block liegen aber nicht alle Intuition-Programmteile.

Was noch kommt und kommen kann

Obwohl durch das Festhalten des Kickstart auf ROM-Bausteinen ein Teil der Flexibilität des Amiga-Systems verlorengegangen ist, haben wir trotzdem nicht das Ende eingeläutet.

Das Betriebssystem ist, wie Sie im folgenden noch sehen werden, immer noch so flexibel aufgebaut, daß es jederzeit von Commodore erweitert werden kann. Aber auch andere Firmen oder Privatpersonen sind dazu in der Lage. Selbst Sie können, wenn Sie Lust haben, dem Amiga zu neuen Fähigkeiten verhelfen.

In jedem Fall in Aussicht stehen aber einige Verbesserungen von Commodore. So ist Ende 1988 das Kickstart 1.3 herausgekommen, das wir während unserer Arbeit zum Buch schon benutzen konnten. Dieses neue Betriebssystem enthält weniger Fehler, ist schneller, weil es größtenteils in Assembler programmiert wurde, und ermöglicht das Booten von anderen Einheiten als nur dem Laufwerk DF0. Sie können beispielsweise durchaus von der neuen mitgelieferten reset-festen RAM-Disk booten oder sogar von einer Festplatte.

Schon angekündigt, aber noch in der Entwicklungsphase ist Kickstart 1.4, das wiederum mehr können soll. Es wird von Grund auf neu programmiert, was aber nicht bedeutet, daß es nicht mehr mit alten Programmen zusammen läuft. Vielmehr sind damit elementare Verbesserungen möglich. Es wird ein MByte-Chip-RAM geben, womit Sie nicht nur mehr Grafik-, sondern auch Sound-Speicher haben. Die Grafik selbst, wird um Multi-Color-Fonts ergänzt. Auch wesentlich bessere Auflösungen gehören zu den Features des neuen Betriebssystems.

8.3.1 Die verschiedenen Modulgruppen

Libraries

Die erste und wohl auch bekannteste Modulgruppe des Betriebssystems sind die Libraries. Library heißt auf Deutsch übersetzt nichts anderes als Bibliothek. Nun sind in einer Bibliothek (geordnet nach Sachgebieten) verschiedene Bücher und Quellen zu finden. In einer Library des Amiga haben wir nur ein Sachgebiet, damit es nicht zu unübersichtlich wird. Zusammengefaßt und gesammelt sind in einer Library nun nicht Bücher, sondern Unterprogramme.

Wir finden in jeder Library zu einem Aufgabengebiet eine Ansammlung von Unterroutinen. Das Besondere und sehr Wichtige ist an einer Library, daß sie ständig ergänzt und verbessert werden kann. Man hat sich darauf geeinigt, daß die Funktionen, die einmal in eine Library aufgenommen wurden, immer so

bestehen bleiben. An ihrer Arbeit darf sich nichts verändern. Denn nur so kann gewährleistet sein, daß auch bei späteren Versionen die alten Programme noch laufen.

Es gibt aber durchaus die Möglichkeit, daß von Commodore ein oder mehrere Unterprogramme neu in eine Library aufgenommen werden. Damit an dieser Stelle jedes Programm weiß, welche Funktionen es ansprechen kann und welche nicht, wurde die Versionsnummer eingeführt. Man kann also sicher sein, daß ein bestimmtes Unterprogramm ab einer festgelegten Versionsnummer immer vorhanden ist. Diese muß nur am Anfang nachgeprüft werden.

Wie ist nun eine Library aufgebaut? Da haben wir eine Menge von Unterroutinen, die alle zum selben Themengebiet gehören. Diese sind zuerst z.B. alphabetisch nach ihrem Namen zu ordnen. Dann wird eine Tabelle aufgestellt. Betrachtet man nun die Library im Speicher, so finden wir zu Anfang immer diese Tabelle. In ihr sind einfach nur die Anfangsadressen jeder Unterroutine festgehalten. Bei einem Aufruf dieser Routinen wird nun nicht die Position der Routine selbst angesprungen, sondern nur der festgelegte Punkt in der Tabelle. Hier steht, wo sich die Unterroutine befindet. Somit kann die Liste der Routinen ständig ergänzt werden, ohne daß man die Sprungbefehle verändern muß. Praktisch!

So eine Library, wie wir sie hier beschrieben haben, kann von jedem Programm, das auf dem Amiga läuft, genutzt werden. Allerdings sind dazu einige Vorbereitungen zu treffen. Man kann sich vorstellen, daß ein großes Chaos herrschen würde, wenn alle gleichzeitig laufenden Programme auf eine Library zugreifen würden. Deshalb meldet man zuerst seinen Zugriff an. Dann bekommt man einen Zeiger zurück, unter dessen Verwendung es ein Kinderspiel ist, die Funktionen anzuspringen. Man gibt lediglich immer die relative Position unseres Tabellenpunktes an, und schon wird die Unterroutine aufgerufen.

Was für Libraries gibt es überhaupt? Der Amiga stellt über ein Dutzend Libraries zur Verfügung. Man unterscheidet hierbei zwischen zwei Typen. Die einen sind so wichtig, weil sie immer gebraucht werden, daß sie in das Kickstart-ROM aufgenommen wurden. Die anderen werden eher selten gebraucht und sind deshalb auf der Diskette. Zu den Gebieten der Libraries zählen die Grafikausgabe, die Diskettenoperationen, die Speicherverwaltung, das Regeln der Windows und Screens und einige andere Verwaltungsaufgaben, die aber nicht sehr anschaulich sind.

Besonders klug ausgetüftelt ist der Umstand, daß die auf der Diskette vorhandenen Libraries natürlich nicht beschränkt sind. Das heißt für den Programmierer, daß es praktisch jederzeit möglich ist, eine neue Library mit neuen Aufgaben zu kreieren, die voll in das Betriebssystem unseres Amiga integriert werden kann. Hier zeigt sich wieder, mit welcher Konsequenz die Entwickler des Amiga daran gedacht haben, ihn offen zu konzipieren.

Devices

Eine ganz andere Modulgruppe des Betriebssystems sind die Devices. Was die Libraries als Sammlung von Unterroutinen waren, sind die Devices als Sammlung von Ein- und Ausgabe-Geräten. Der Amiga arbeitet hier anders als die kleinen Home-Computer.

Die Ein- bzw. Ausgabe läuft hier nicht direkt über Datenkanäle, die schon über die Hardware festgelegt sind. Es gibt also nicht eine feste Leitung, auf der die Grafiken und Texte an den Drucker geschickt werden. Beim Amiga gibt es ein Device, das die Daten für den Drucker empfängt und an ihn weiterleitet. Das Device ist ein Programm, das sich zwischen den Datentransport klemmt. Dadurch haben wir einige Vorteile.

Zuerst der Vorteil, daß es sich um ein Programm handelt. Es ist dadurch fast beliebig oft möglich, neue Devices für neue Aufgaben zu kreieren. Es gibt ein Device zur Druckeransteuerung, jeweils eins zur Ein- und Ausgabe über den seriellen und parallelen Port und noch einige mehr. Aber wenn ein neues Device gebraucht wird, schreibt der Programmierer sich einfach eins!

Dazu gibt man ihm einen neuen Namen und stellt alle wichtigen Parameter ein. Ab sofort, und zwar, wenn das Device in das System eingebunden ist, können Daten an dieses gesendet oder von ihm empfangen werden. Wie die Daten verarbeitet, ausgewertet oder weitergegeben werden, hängt ganz vom Device ab. Genau wie auch Libraries ist ein Device aber nicht sofort nach seinem Entstehen ansprechbar. Es muß zuerst möglichst beim Start des Systems, also in der Startup-Sequence, eingebunden werden. Dann können wir es, ähnlich wie DF0: oder PRT:, ansprechen und ihm Daten zuschicken oder von ihm anfordern.

Was die Devices aber so besonders macht, ist die Tatsache, daß sie die neuen Daten, die an sie geschickt werden, weiterverarbeiten, bevor sie sie an das endgültige Zielgerät schicken. So hat das Printer-Device die Möglichkeit, anhand der Einstellungen im Preferences-Dräcker-Menü die Daten anzupassen und erst dann an den Drucker zu schicken.

Handler

Alle Devices werden von dem AmigaDOS verwaltet. DOS steht hier für Device Operation System. Es ist sozusagen ein Aufsichtsprogramm, das bei jedem Versuch, ein Device anzusprechen, die Leitung übernimmt. Ein Device ist danach betrachtet kein wirkliches Programm. Es existiert nur als Eintrag in einer Tabelle.

Das Programm, das die Arbeit bei der Datenübermittlung übernimmt ist der Handler. Er ist auch in der Tabelle eingetragen, in der alle Eigenschaften des Devices vermerkt sind. Übergeben wir nun Daten an das Device, werden diese zum Handler geschickt, der sie auswertet, bearbeitet und dann letztendlich an das bestimmte Ziel weiterleitet. So werden beim Schicken von Texten einer Textverarbeitung zum Drucker diese in Wirklichkeit an das Device PRT: gesendet. Dort bearbeitet ein Handler das Empfangene. Steuersequenzen werden nach den Einstellungen im Preferences-Programm umgesetzt und nach den Daten in der Printer-Information dekodiert. Am Ende kann dadurch ein ganz anderer Text entstanden sein, der nun wirklich an den Drucker geschickt wird. Auch hier geht es noch einmal darum, was eingestellt wurde. Denn einen Drucker können Sie sowohl am seriellen wie auch am parallelen Port des Amiga anschließen. Also untersucht der Handler noch einmal, wohin die Daten gesendet werden müssen.

Obwohl das Printer-Device noch ziemlich einfach ist (die Daten laufen hier nur in einer Richtung) zeigt es, wie kompliziert die Vorgänge innerhalb des Betriebssystems sind. An dieser Stelle wollen wir uns einmal ansehen, wie die einzelnen Ebenen des Amiga untereinander verknüpft sind. Bleiben wir hier ruhig bei der Textverarbeitung, die ihren Text ausdrucken soll.

Im Programm selbst wird mit Hilfe der Betriebssystem-Funktionen ein Text erstellt. Diese Funktionen sind alle aus den Libraries genommen. Mit der Graphics.Library wird der Text in verschiedenen Fonts ausgegeben, die Intui-

tion.Library sorgt für die richtige Verständigung mit dem Benutzer, und die Exec.Library hilft uns nun, die Daten - nichts anderes ist unser Text - an das Printer.Device zu schicken. Dieses gibt sie über die Hardware weiter an die Schnittstelle, die sie dem Drucker übermittelt.

Exec - der Verwalter

Die größte und wichtigste Rolle bei dieser sehr aufwendigen Organisation des Betriebssystems spielt Exec. Exec ist eine Library wie jede andere auch. Mit einer Ausnahme! Sie ist die einzige Library, die ständig offen, d.h. für jeden zugänglich ist. Ihre Basisadresse stellt ein Unikum dar, denn sie ist als einzige Basisadresse aller Elemente des Betriebssystems fest in einer Speicheradresse des Amiga abgelegt. Die Rede ist von der Speicherstelle 0x00000004, in der immer ein Zeiger auf die ExecBase, die Basisadresse der Exec.Library, liegt.

Die Exec.Library selbst ist alleine für Verwaltungsaufgaben geschaffen. Allerdings ist diese Verwaltungsarbeit so elementar, daß ihr eine besondere Position zukommt. Exec verwaltet neben dem Speicher beliebige Arten von Listen, die Libraries selbst, die Nachrichtenübertragung im System, jede Ein- und Ausgabe, die Interrupts und das Multitasking. Um das Verständnis vom Aufbau des Betriebssystems zu erhöhen, sehen wir uns die einzelnen Elemente etwas näher an.

Da wäre als erstes die Speicherverteilung. Nach dem Einschalten des Amiga prüft dieser, wieviel Speicher ihm zur Verfügung steht. Im Normalfall sollten dies mindesten 512 KByte sein. Es ist aber durchaus möglich, daß Speichererweiterungen angeschlossen sind. Diese werden - vorausgesetzt, sie sind autokonfigurierend - gleich beim Einschalten als vorhanden erkannt. Nun weiß die Exec.Library also, wieviel Speicher ihr zur Verfügung steht. Kommen wir mit einem Befehl, der Exec anweist, uns Speicher zuzuteilen, sieht dieses in seiner Liste nach, ob ein Speicherstück in der gewünschten Größe zu besorgen ist. Bei einer positiven Antwort erhalten wir als Ergebnis die Adresse dieses Speichers, und Exec notiert sich den Startpunkt und die Größe. Im anderen Fall erhalten wir eine Fehlermeldung. Auf die gleiche Weise gehen nun auch alle anderen Programm vor. Jeder holt sich seinen Speicher über die Verwaltungsroutinen der Exec.Library. Wird dann ein Speicherbereich nicht mehr benötigt, sollte man dies Exec mitteilten. Er wird daraufhin sofort aus der Belegliste gestrichen und ist nun wieder jederman zugänglich. Eine

einfache Arbeit, die aber gemacht werden muß bei einem Computer, der es erlaubt, beliebig viele Programme quasi gleichzeitig laufen zu lassen.

Zu diesem Zweck der Speicherverwaltung benutzt Exec eine Form der Liste, wie sie auch für andere Dinge benutzt wird. Eine Liste ist einfach eine Struktur, in der die benötigten Daten gespeichert sind und die jeweils auf das nächste und vorhergehende Listenelement zeigt. Nur so ist es möglich, ein Element aus der Liste zu streichen. Aber für alle diese Operationen besitzt Exec eigene Funktionen. Diese werden alle über die Exec.Library aufgerufen, so daß Sie sich - selbst wenn Sie so etwas programmieren sollten - keine großen Gedanken über die Organisation machen müssen.

Diese organisatorischen Aufgaben bewältigt Exec nicht nur auf dem Gebiet der Speicherverwaltung. Durch diese Library wird es erst möglich, den Multitasking-Betrieb, d.h. mehrere Programme gleichzeitig, laufen zu lassen. Dafür verfährt man so:

Auch für die laufenden Programme führt Exec ein Liste, die natürlich nach dem Einschalten noch leer ist. Wird jetzt ein Programm in den Speicher geladen, wofür die oben besprochene Speicherverwaltung benutzt wird, trägt Exec dieses Programm mit seinem Namen und seiner Priorität in die Programm-Liste ein. Auch jedes weitere Programm wird so in diese Liste geschrieben. Wenn der Prozessor nun arbeitet, verteilt Exec je nach Priorität Arbeitszeit des Prozessors an die Programme. Vorausgesetzt, daß alle Programme die gleiche Vorrangigkeit besitzen, sieht das etwa so aus: Das erste Programm darf seine ersten 10 Befehle abarbeiten und wird dann in den Warte-Modus gesetzt. Daraufhin folgt das nächste Programm in der Liste, das auch seine ersten 10 Befehle ausführen kann. Und so geht das immer weiter, bis alle Programme einmal versorgt wurden. Danach geht es wieder von Anfang an los.

Gibt es nun ein Programm, das eine höhere Priorität besitzt, so werden ihm mehr Befehle gestattet, die es bei einem Durchlauf bearbeiten kann. Das erste Programm würde dann z.B. 14 Befehle ausführen, während alle anderen nur noch 8 Befehle abarbeiten können. Exec berechnet also je nach Priorität die Anzahl der Befehle, die innerhalb eines Durchlaufs von jedem Programm an den Prozessor geschickt werden können. In der Liste der Programme gibt es zu jedem Zeitpunkt aber immer nur ein Programm, das seine Befehle an den Prozessor schickt, alle anderen warten.

DOS - Device Operation System

Einen wichtigen Teil im Betriebssystem nimmt das AmigaDOS ein. Um eine möglichst hohe Kompatibilität zu gewährleisten, hat man ein internes Geräte-System eingeführt, bei dem sich jeder Daten-Empfänger oder -Sender als Gerät deklarieren kann. Aber beginnen wir mit einem Beispiel aus dem Alltag.

Wenn Sie ein Programm auf die Diskette schreiben, lautet der Befehl etwa so:

```
SAVE "DF0:Programme/Test2"
```

Sie haben damit ihr Programm, das den Namen Test2 trägt, auf die Diskette im Laufwerk 0 geschrieben, und zwar in das Verzeichnis Programme. Man kann das ganze aber auch anders betrachten, systemnäher. Dann würde man es so deuten: Sie haben an das Device (engl. für Gerät) DF0: Daten geschickt, nämlich Ihr Programm, mit der Anweisung Programme/Test2. Das hört sich etwas verdreht an, entspricht aber der System-Deutung ihres Befehls.

Ohne Zweifel ist, daß wir Daten verschicken. Diese Daten sind hier der Programmtext. Er wird an das Gerät DF0: geschickt und gleichzeitig teilen wir dem Gerät noch mit, wo es die Daten genau ablegen soll. Die Floppy oder vielmehr ihr Verwaltungssystem interpretiert das als Verzeichnis- und Programm-Angabe. Um zu zeigen, daß diese Geräte wie z.B. DF0: frei untereinander austauschbar sind, versuchen Sie es ruhig einmal damit, ein Programm auf PRT: - ohne weitere Angaben - zu speichern. Es funktioniert!

```
SAVE "PTR:",a
```

Es ist also egal, ob Sie LLIST oder diesen SAVE-Befehl eingeben. Die Wirkung ist die gleiche, weil beide Befehle den Amiga anweisen, die Daten an den Drucker zu senden.

Der elementare Unterschied zwischen den beiden Geräten ist aber, daß von der Diskettenstation die gespeicherten Programme wieder geladen werden können. Vom Drucker ist das so gut wie unmöglich. Es gibt also ganz allgemein zwei Arten der Devices: Sogenannte In/Out-Devices wie Diskettenlaufwerke oder nur Out-Devices wie z.B. der Drucker. Theoretisch wäre auch ein nur In-Device denkbar, das man aber auf dem Amgia noch immer nicht standardmäßig realisiert hat. Ich denke hier besonders an eine Meßstation, die

immer Daten liefert, aber keine Befehle oder Daten empfangen kann. Eine praktische Anwendung ist z.B. ein Sound Digitizer.

Für die Verwaltung dieser ganzen Geräte ist nun das AmigaDOS zuständig. Es führt eine Tabelle, in der alle Geräte aufgeführt sind. Zu jedem Gerät gibt es neben seinem Namen noch einige Daten, die bestimmen, welche Eigenschaften es hat. Außerdem führt das DOS eine Liste der Programme, die die Daten und Befehle eines Devices verwalten. Bei den Floppies ist das der File-Handler, der anhand der Angaben hinter dem Device-Namen die Verzeichnis-Struktur einer Diskette, einer RAM-Disk oder einer Festplatte erkennt. Für jedes dieser Systeme benötigen Sie übrigens unterschiedliche Handler, weil jeder andere Hardware-Anforderungen stellt. Nur für den Benutzer erscheint es gleich, wenn er ein File auf der Diskette, der RAM-Disk oder einer Festplatte sichert. In Wirklichkeit aber unterscheiden sich die Vorgänge wesentlich.

Zum Speichern eines Files auf der Diskette muß zuerst festgestellt werden, welche Blöcke noch unbelegt sind. Hierher werden die Daten über die Ein-/Ausgabe-Bausteine transferiert und gespeichert. Zusätzlich wird noch ein Info-Block angelegt, mit dem man später den Eintrag im Inhaltsverzeichnis wiederfindet. Auf der RAM-Disk muß zuerst aus dem vorhandenen Speicher ein Block besorgt werden, der für das File ausreicht. Dann werden hierher über den Blitter die Programmdaten im Speicher verschoben, und ein Eintrag ähnlich des Disketteneintrags wird angelegt. Auch auf der Festplatte wird wie bei den Disketten nach einem oder mehreren freien Blöcken gesucht. Doch hier ist die Hardware-Organisation wieder anders, weil wir es mit einer größeren Anzahl von Spuren und Seiten zu tun haben. Man stellt also fest, daß sich alle Vorgänge ähneln, aber leider nicht gleich sind, so daß man mit einem Verwaltungs-Programm nicht auskommt.

Intuition als Unterstützung zur Workbench

Beim Entwickeln der Bedieneroberfläche des Amiga waren die anderen Oberflächen den Programmierern weitgehend bekannt. Die IBM-Oberfläche mit der Symbol- und Window-Verwaltung und auch die Atari-GEM-Oberfläche mit ihrer begrenzten Anzahl von Windows waren eine Voraussetzung, an der man sich messen wollte.

Der erste Grundsatz der Planung war von Anfang an klar. Es durfte auf keinen Fall irgendwelche Beschränkungen geben, die die Bedieneroberfläche unattraktiv machen oder sie vor späteren Verbesserungen abhalten würden. Damit ging man von der gleichen Grundlage aus:

Es sollte eine Fläche werden, die Workbench, auf der man arbeitet wie auf dem normalen Schreibtisch auch. Es gibt Symbole, die bestimmte Gegenstände repräsentieren, und es wird Windows geben, die Aufgabenbereiche von anderen abgrenzen. Auch die Menüs dürfen nicht fehlen. Alles mit der Auflage: keine Begrenzungen. Man machte sich an die Entwicklung. Angelehnt an die Graphics.Library, wurde zuerst eine Layers.Library geschaffen, die in der Lage ist, grafische Bereiche getrennt von anderen zu verwalten. So können Flächen-Überlagerungen stattfinden, die bei allen Grafik-Befehlen berücksichtigt werden. Überschneiden sich zwei dieser Flächen, so werden die Grafik-Befehle, die den verdeckten Bereich betreffen, nicht optisch ausgeführt, sondern in einem Speicher abgewickelt, in dem sich die Zwischenspeicherung des Verdeckten befindet.

Aus diesen Flächen entstanden die Windows, die eine Weiterentwicklung darstellen. Die Flächen waren zu Anfang nur rechnerisch da und wurden erst lebendig und sichtbar, wenn man in ihnen zeichnete. Solange kann der Benutzer aber nicht warten, und deshalb gab es einen Window-Rand, der kennzeichnet, wo ein Bereich endet und ein neuer beginnt. Damit der Benutzer keine Schwierigkeiten bei seiner Arbeit bekommt, versuchte man, ihm Komfort zu verschaffen. Die Flächen wurden bewegbar und konnten in ihrer Größe eingestellt werden. Nun fehlte nur noch eins. War einmal ein wichtiges Window im Hintergrund, so brauchte man große Anstrengungen, um dies sichtbar zu machen. Die verdeckenden Windows mußten verschoben und verkleinert werden, bis alles zu sehen war. Mit den Gadgets zum Verändern der Reihenfolge, in der die Windows aufeinander liegen, ist auch das eine Leichtigkeit.

Nun war es nötig, bestimmte Bereiche zu schaffen, die Aktionen auslösen, wenn man dorthin mit der Maus klickt. Die Gadgets waren geschaffen. Schon bei den Windows eingesetzt, gibt es sie jetzt auch für den Benutzer und Programmierer. Er kann festlegen, welcher Bereich für einen Mausklick sensibel ist und auf welche Art. Das Programm bekommt dann eine Nachricht, sozusagen, wenn der Amiga Schmerzen spürt, und kann darauf reagieren. Diese Gadgets werden nicht nur in Windows benutzt, auch die Workbench verwendet diese Objekte zur Darstellung aller Symbole für Disketten, Verzeichnisse, Programme und Daten-Files.

Die letzte große Ergänzung erfuhr Intuition durch die Menüs, die eine Kombination aus Gadgets und Windows sind. Und zwar wird durch das Klicken auf die rechte Maustaste der Menümodus von Intuition aktiviert. Ab jetzt gilt die obere Screen-Leiste als empfindlicher Bereich. Fährt man einen der markierten Punkte an, rollt sofort ein Pulldown-Menü herunter. Diese Menüs sind nichts anderes als Windows, in denen verschiedene Funktions-Namen aufgelistet sind. Doch anders als bei den Gadgets, die man extra aktivieren mußte, sind bei den Menüs die Punkte schon aktiv, wenn nur die Maus über ihnen steht. Man wählt hier auch nicht dadurch aus, daß man den Punkt durch Niederdrücken der Maustaste aktiviert, sondern man muß die Taste loslassen, um eine Aktion zu bestätigen. Damit haben wir eine komfortable Bedieneroberfläche erhalten.

8.3.2 Die Ergänzungen zum Grundstock

Mit den wichtigen Libraries des Betriebssystems ist der Grundstock fest geschüttet: die Grafik wird von der Graphics.Library erledigt, und die Dos.Library kümmert sich um alle Ein- und Ausgabe-Operationen. Wenn es um Verwaltung geht, tritt Exec in Aktion, und bei Fragen der Benutzerführung hilft uns Intuition. Dabei dürfen wir aber nicht die anderen Libraries vergessen, die helfend zur Hand gehen, wenn es um Aufgaben geht, die nicht sehr häufig oder selten ausgeführt werden müssen.

So könnte Intuition überhaupt nicht arbeiten, wenn nicht die Layers.Library dabei Unterstützung gewähren würde. Denn nicht Intuition verschiebt die Windows, das macht die Layers.Library genauso, wie auch die aufwendigen Arbeiten, wenn sich zwei Arbeitsflächen überdecken. Es gibt, wie man sieht, noch einige sogenannte Hilfs-Libraries, auf die unsere großen vier aufbauen.

Die Graphics.Library ist zwar in ihrer Arbeit selbständig, wird aber durch die Diskfont.Library unterstützt. Die Funktionen, die sich auf Zeichensätze beziehen, arbeiten mit allen ihrer Art. Man kann nun auch einen von der Diskette nachladen - mit der Diskfont.Library - und darauf die Funktionen der Graphics.Library anwenden. Auch das geht.

Ein anderes wichtiges Teilgebiet sind die mathematischen Berechnungen. Sie sind der eigentliche Grund, weswegen man den Computer erfunden hat. Auch der Amiga kann natürlich rechnen. Und zu dieser Fähigkeit verhilft ihm wie-

der eine Library: die Mathematik-Library. Es sind in diesem Fall sogar mehrere, weil man zwischen verschiedenen Berechnungen unterscheidet. Es gibt einmal die normalen Grundrechenarten mit den Erweiterungen wie z.B. Potenzen oder Wurzeln. Außerdem gibt es aber auch noch die trigonometrischen Funktionen, die zusätzlich in einer separaten Library untergebracht sind. Alles dies wurde dann noch in verschiedenen Zahlenformaten angeboten, damit der Programmierer gleich wählen kann, ob er einfach, doppelt oder viermal so genau rechnen will, wie es angeboten wird. Davon hängt natürlich auch die Geschwindigkeit ab, in der die Berechnungen durchgeführt werden.

Wie Sie sehen, zeigt der Amiga Farbe! Neben den sturen Verwaltungsaufgaben finden sich durchaus Libraries, die auf spezielle Wünsche eingehen. Es ist dabei zu bedenken, daß wir hier nicht auf jede Library eingehen konnten. Schon mit der Version 1.3 ist eine neue hinzugekommen, und dies kann sich weiter fortsetzen. Auch Sie können eine Library schreiben und dem Betriebssystem ergänzen!

Es ist nun aber an der Zeit, auf die Dinge hinzuweisen, die den Amiga so individuell machen. Jeder Computer kann Texte bearbeiten, Grafiken zeichnen und Töne ausgeben. Aber es gibt etwas, das den Amiga von anderen "normalen" Computern deutlich unterscheidet. Die Rede ist nicht von einer bahnbrechenden Erfindung, die das Computer-Leben revolutionieren wird, sondern von einer einfachen und kleinen Möglichkeit, die aber an fast jeder Stelle geboten wird und so die ideale Arbeitsumgebung schafft.

Sie werden es längst bemerkt haben. Wir reden von den kleinen Einstellungen, die man - Gott sei Dank - in fast jeder Situation machen kann, bei der man mit dem Amiga arbeitet. Wir haben uns dazu drei Beispielthemen herausgegriffen.

Keymaps

Der Amiga ist flexibel, wie es sich immer wieder in seinem Betriebssystem ausdrückt. Nichts ist fest, damit jederzeit Veränderungen gemacht werden können. Davon profitieren auch wir.

So ist z.B. die Tastatur auf die amerikanische Version des Amiga ausgerichtet, was sicherlich auch verständlich ist, denn dort wurde er schließlich ent-

wickelt. Allerdings war man gleich von Anfang an der Meinung, daß solch ein Computer auch in anderen Ländern der Welt vertrieben werden muß. Deshalb wurden solche landesabhängigen Einstellungen wie z.B. die Tastaturbelegung variabel gehalten.

Fest eingebaut ist die amerikanische Belegung der Tasten. Aber mit wenig Aufwand läßt sich daraus eine deutsche Tastatur machen, wenn man nur einen entsprechenden Befehl im Betriebssystem oder CLI aufruft. Das Keyboard. Device, das dafür zuständig ist, lädt einfach die neue Tastaturtabelle in die nötigen Register und übermittelt ab jetzt bei einem Tastendruck das deutsche Zeichen.

Fonts

Ähnlich ist es auch mit den Zeichensätzen. Wie Sie bestimmt bemerkt haben, gibt es beim Amiga keinen festen Zeichensatz, in dem alle Texte ausgegeben werden. Er kann bei jeder Ausgabe neu gewählt werden. Doch es muß einen Zeichensatz geben, der sozusagen als Standard im Computer vorhanden ist.

Deshalb wurde in das Kickstart der Topaz-Zeichensatz integriert. Er existiert dort in zwei Schriftgrößen. Topaz 8 schafft bis zu 80 Zeichen pro Zeile und ist mit einem Monitor gut zu lesen. Für diejenigen, die einen Fernseher angeschlossen haben, gibt es Topaz 9, der etwa 60 Zeichen pro Zeile darstellen kann.

Aber auch hier ist das System flexibel aufgebaut. Möchten Sie einen anderen Zeichensatz, so können Sie diesen natürlich jederzeit nachladen und mit Hilfe der Graphics.- oder Diskfont.Library einsetzen. Ab sofort werden die Ausgaben in dem neuen Zeichensatz erledigt. Das ist gar kein Problem.

Anhang

Anhang A: Grundlagen und weiterführende Literatur

Grafik:

Das neue Supergrafik-Buch
DATA BECKER
Merowingerstr. 30
4000 Düsseldorf 1

Hardware:

Amiga Intern
DATA BECKER
Merowingerstr. 30
4000 Düsseldorf 1

Amiga Hardware Reference Manual
Commodore Business Machines, Inc.
Addison Wesley

Programmierung:

Das große C Buch zum Amiga
Das große AmigaBAISC Buch
Maschinensprache für Einsteiger
Das große AmigaDOS Buch

alle DATA BECKER

Anhang B: Fachwortlexikon für den Aufsteiger

In diesem kleinen Fachwortlexikon finden Sie alle in diesem Buch verwendeten Begriffe, von denen wir meinen, daß sie noch einmal gesondert erklärt werden sollten. So können Sie nicht nur beim Lesen dieses Buches, wenn Ihnen ein Begriff unklar ist, hier nachschlagen, sondern auch, wenn Sie irgendwo anders z.B. in einer Fachzeitschrift ein Fachwort entdecken, daß Ihnen unbekannt ist.

Um alle eingetragenen Begriffe schneller zu finden, haben wir zuvor noch ein kleines Inhaltsverzeichnis gestellt, in dem Sie die Seitenzahl finden können.

Inhaltsübersicht

Alert ... 520
AmigaDOS ... 520
ASCII ... 521
Assembler ... 521

BASIC ... 522
Benutzeroberfläche ... 522
Betriebssystem ... 523
Binäres Zahlensystem ... 523
Bit ... 523
Booten ... 523
Byte ... 524
Button ... 524

C ... 524
CLI ... 524
Compiler ... 525
Cursor ... 525

Datei ... 525
Datenverwaltung ... 526
Device ... 526
Directory ... 526
Diskette ... 526

DOS 527
Dualsystem 527

Editor 527

Festplatte 528
File 528
Floppy 529
Font 529

Gadget 529
Guru-Meditation 529

Hardware 530

Icon 530
IFF 530
Inhaltsverzeichnis 531
Interpreter 531
Intuition 531

Joystick 531

Kickstart 531

Library 532

Maschinensprache 532
Maus 532
Menü 533
Multitasking 533

Norm 533

Patch 534
Programmiersprache 534
Prozessor 535

RAM 535
RAM-Disk 536

Requester 536
Reset 536
ROM 537

Screen 537
Software 537
Standard 538

Tastatur 538

Verzeichnis 539
Virus 539

Window 539
Workbench 540

Zeichensatz 540
Zeileneditor 540

Alert

Der Alert ist die stärkere Form des → Requesters. Er ist im Grunde auch eine Meldung an den Benutzer, die mit einer Nachfrage verbunden ist. Allerdings ist der Anlaß, weswegen ein Alert gewählt wurde, wesentlich wichtiger und dringender. Ein Alert kennzeichnet meistens eine ausweglose Situation, die nur noch dem Benutzer mitgeteilt werden muß (siehe → Guru-Meditation). Der Alert wartet in diesem Fall nur auf eine Bestätigung, die ihm die Sicherheit gibt, daß der Anwender von der Meldung Notiz genommen hat.

AmigaDOS

Das AmigaDOS ist ein Teil unseres → Betriebssystems. Es ist wie alle anderen → Libraries auch im → Kickstart-ROM untergebracht. AmigaDOS ist die Abkürzung für Amiga Device Operation System und heißt auf Deutsch, daß es das Geräte-Verwaltungs-

System ist. AmigaDOS managed also alle Ein- und Ausgabe-Geräte, die man am Amiga betreiben kann. Damit sind aber nicht nur Diskettenlaufwerke und Drucker gemeint. Auch die Bildschirmausgabe kann vom AmigaDOS abgewickelt werden.

Zu jedem Gerät benutzt AmigaDOS ein sogenanntes Device, das uns unter einem kurzen Namen bekannt ist. Die Diskettenlaufwerke tragen die Bezeichnungen DF0: bis DF3:, jeder angeschlossene Drucker den Namen PTR: und die Schittstellen SER: oder PAR:. Wir können in dieser Art jederzeit neue Geräte kreieren, wie es z.B. bei Version 1.3 der Workbench geschehen ist, wo in der MountList, über die man neue Geräte definiert, z.B. das NewCON:-Device, der Speaker oder die Pipe hinzugekommen sind.

Wenn Sie wollen, können Sie sich auch eigene Geräte über die Disketten schaffen. Mit dem CLI-Befehl Assign ist das ohne weiteres gestattet.

ASCII

American Standard Code of International Interchange: Wie sich schon aus der Erklärung der englischen Abkürzung ersehen läßt, ist mit der Bezeichnung ASCII ein amerikanischer Standard-Code gemeint, der für den internationalen Datenaustausch vorgesehen ist.

Assembler

Im Gegensatz zu anderen → Programmiersprachen ist Assembler eine sehr maschinennahe Sprache. Auch mit ihr kann der Amiga programmiert werden. Allerdings zeichnet sich Assembler dadurch aus, daß nicht wie in → BASIC einfache, aber komplexe Anweisungen gegeben werden, sondern wir kleinschrittig dem Prozessor direkt jeden

Befehl geben müssen. Das hat zur Folge, daß ein Programm zum Ausgeben eines Textes in BASIC nur eine Zeile, in Assembler aber fast eine Seite annehmen kann. Allerdings sind Assembler-Programme im Vergleich zu BASIC oder anderen höheren Sprachen wie → C oder Pascal wesentlich schneller.

BASIC

BASIC ist eine der populärsten Computer-Sprachen der Welt. Der Name ist die Abkürzung für Beginners Allpurpose Symbolic Instruction Code und heißt soviel wie "Ein Symbolischer Anweisungs-Text für Anfänger", womit die leichte Verständlichkeit gemeint ist, die durch die der englischen Sprache nachempfundenen Befehlsworte entsteht. Diese → Programmiersprache wird auf dem Amiga kostenlos mitgeliefert und befindet sich auf der Extras-Diskette. Es handelt sich bei BASIC um eine → Interpreter-Sprache im Gegensatz zu C, das eine → Compiler-Sprache ist.

Benutzeroberfläche

Ist der allgemeine Begriff, mit dem man die Gestaltung bezeichnet, über die man Daten und Befehle an den Computer weiterleitet. Bei vielen Computern ist die Benutzeroberfläche nur ein → Zeileneditor, mit dem immer ein Befehl pro Zeile eingegeben wird. Es gibt auf diesem Gebiet dann aber verbesserte Weiterentwicklungen, die es schon erlauben, mehrere Zeilen gleichzeitig zu bearbeiten.

Der Amiga hingegen besitzt eine wesentlich bessere Benutzeroberfläche. Sie gestattet es, ohne die Eingabe von Text Befehle zu geben oder Operationen auszuführen. Das macht es besonders für den Anfänger einfach, den Amiga zu bedienen. Die → Workbench, so nennt man die Arbeitsfläche, die nach dem

Einschalten erscheint, nutzt alle diese Funktionen aus. Als Grundlage für die Gestaltung der Benutzeroberfläche wurde eine Fensterverwaltung benutzt, die von → Intuition gemanaged wird.

Betriebssystem

Das Betriebssystem des Amgia 500 ist im → Kickstart-ROM zu finden. Es enthält alle wichtigen Steuerprogramme, die die Grundaufgaben erledigen. Gleich zu Anfang wurde es so ausgelegt, daß es jederzeit fast beliebig erweiterbar ist. Ein wichtiger Bestandteil des Betriebssystems ist das AmigaDOS, das die gesamte Ein- und Ausgabe steuert.

Binäres Zahlensystem

siehe → Dualsystem

Bit

Bit ist die kleinste Recheneinheit, mit der ein elektronischer Rechner arbeiten kann. Man vergleicht diese auch meist mit dem Ein- und Ausschalten einer Glühlampe, die ja entweder nur brennen kann oder erlischt. Aus mehreren (8) Bits wird ein → Byte zusammengesetzt. Dies ist die Masse an Bits, die ein 8-Bit-Prozessor verarbeiten kann. Der → Prozessor des Amiga kann gleichzeitig mit 16 Bits arbeiten und sogar 32 Bit simulieren, was seine Rechengeschwindigkeit erhöht.

Booten

Als Booten bezeichnet man den Vorgang nach dem Einschalten oder einem → Reset, wenn von der Workbench-Diskette die wichtigen Informationen für den Start des Systems gelesen werden. Diese befinden sich auf der Diskette im Boot-Block, der auch ein Programm enthalten kann. Dieses wird oft bei Spielen benutzt, um gleich nach dem Einlegen der Diskette eine Grafik auf den Bildschirm zu bringen.

Byte

Ein Byte besteht aus 8 → Bits und stellt die normierte Recheneinheit eines Microcomputers dar. Alle Bits werden von 0 bis 7 durchnumeriert und stehen für die Hochzahl, mit der 2 potenziert wird. (Die Zwei ergibt sich aus den beiden Zuständen eines elektischen Leiters: Strom fließt oder fließt nicht). Dadurch ergibt sich ein Zahlenwert von maximal 256 Zuständen, die durch ein Byte dargestellt werden können.

Button

Mit Button bezeichnet man die Knöpfe, die beim Joystick oder der Maus niedergedrückt werden können. Dieser Begriff kommt aus dem Englischen und heißt übersetzt Knopf.

C

C ist der Name einer Programmiersprache, die besonders an die Sprache Pascal angelehnt ist. Bei C handelt es sich um eine Weiterentwicklung von BCPL. Beide Sprachen wurden auch bei der Programmierung des Amiga-Betriebssystems verwendet und sind noch heute die gängigen Sprachen auf dem Amiga neben → BASIC und → Assembler. C-Programme werden nicht über RUN gestartet, wie es bei BASIC der Fall ist, sondern müssen zuerst übersetzt werden, wofür ein → Compiler notwendig ist.

CLI

CLI ist die Abkürzung für Command Line Interface. Damit bezeichnet man die Eingabe-Einheit, mit der die → AmigaDOS-Befehle eingegeben werden können. Das CLI ist hauptsächlich zur Eingabe von Disketten-Befehlen vorgesehen, mit denen die Struktur einer Diskette neu organisiert werden kann. Es eignet sich aber auch als Befehlsebene zum Programmieren, von der ein → Editor und andere Hilfsprogramme aufgerufen werden. Im Gegensatz zur → Workbench ist das CLI

wesentlich differenzierter, weil es genauere Einstellungen erlaubt.

Compiler

Ein Compiler ist ein Programm, das den Programm- → Quellcode (-text) in eine für den Computer verständliche Form bringt. Dabei werden die für den Menschen verständlichen Befehlsworte und Anweisungen in eine sich daraus ergebende Kombination aus → Assembler-Befehlen gesetzt. Der Compiler liefert als Endprodukt ein File mit einem dem Quelltext entsprechenden Assembler-Programm.

Cursor

Zur Eingabe von Texten verwendet man, ähnlich wie bei der Schreibmaschine, die Tastatur. Die Position auf dem Blatt Papier wird aber bei der Schreibmaschine durch den Wagen repräsentiert, was beim Computer auf dem Bildschirm nicht möglich ist. Deshalb hat man den Cursor geschaffen. Es handelt sich um einen Kasten oder einen Strich in der Größe der Zeichen, der immer an der Position zu finden ist, an der neuer Text geschrieben wird.

Es gibt auch den Maus-Cursor, der einfach die Position der Maus auf dem Bildschirm anzeigt. Sie werden den auch als Mauspfeil bezeichneten Cursor sicher kennen.

Datei

Eine Datei ist ganz allgemein eine Ansammlung von Daten, die in irgendeiner Form festgehalten sind. Beim Computer bezeichnet man damit Daten, die auf der Diskette oder einem anderen Datenspeicher gesichert sind. Dies müssen nicht unbedingt Daten aus einer → Datenverwaltung sein, sondern auch Texte oder einfache Zahlenketten bzw. Tabellen gelten als Datei. Sie werden auf der Diskette

als → File gespeichert und sind im → Inhaltsverzeichnis aufgeführt.

Datenverwaltung

Eine Datenverwaltung ist ein spezielles Programm, mit dem man in der Lage ist, Daten beliebiger Anordnung zu gliedern und zu sichern. Die eingegeben Daten können jederzeit unter einem Suchbegriff aufgerufen werden. Je leichter oder je komlexer die Suche ist, desto besser ist die Datenverwaltung einzustufen.

Device

Da es für die Ein- und Ausgabe beim Amiga nicht starr festgelegte Geräte gibt, wurden die Devices entwickelt. Mit Hilfe des → AmigaDOS kann für jede Hardware, die Daten empfängt oder sendet, ein Device im Amiga → Betriebssystem installiert werden. An dieses oder von diesem können wir Daten senden oder empfangen.

Directory

Auf der → Diskette und → Festplatte sind die Daten und Programme in → Dateien als sogenannte → Files abgespeichert. Um diese zu gliedern, wurden Verzeichnisse entworfen, die, immer nach Gruppen sortiert (dies ist dem Benutzer überlassen), Files aufnehmen. Diese Verzeichnisse bezeichnet man als Directories. Jedes Directory kann weitere Unterverzeichnisse enthalten, die wie ein Baum angeordnet sind und eine weitere Gliederung hervorrufen.

Diskette

Disketten sind der bei Computern übliche Datenspeicher. Neben den Kassetten, die schon etwas aus der Mode gekommen sind, gibt es noch die → Festplatten. Sie ermöglichen gegenüber den Disketten einen schnelleren Zugriff mit mehr Speicherkapazität. Das Lesegerät, das für eine Diskette benötigt wird, nennt man Diskettenstation oder → Floppy.

Diskettenstation siehe → Floppy

DOS DOS ist die Abkürzung für Disk Operation System. Damit bezeichnet man den Teil des → Betriebssystems, der für die Verwaltung der Diskettenaufzeichnungen zuständig ist. Auf dem Amiga wurde dieser Begriff ausgedehnt auf die Bezeichnung Device Operation System, weil die Organisation hier nicht über ein Ein- und Ausgabesystem abgewickelt wird, sondern weil für jedes Gebiet ein spezielles Device zur Verfügung steht.

Dualsystem Da der Computer eine mit Strom betriebene Maschine ist, kann er auch nur diesen als Speicher benutzen. Strom erlaubt nun aber nur zwei Zustände zur Speicherung: entweder er fließt oder er fließt nicht. Das hat zur Folge, daß das gesamte Speichersystem darauf ausgelegt werden mußte, mit der Zweier-Möglichkeit, Daten zu speichern. Daher folgt auch der Name Dual- (Zweier-) System. Man setzt nun eine Zahl ähnlich wie beim uns bekannten dekadischen Zahlensystem zusammen. Nur steht jetzt eine Ziffer nicht für eine Zehnerpotenz, sondern für eine Zweierpotenz. Analog dazu haben wir auch nur noch zwei Ziffern (0, 1) anstatt zehn.

Editor Als Editor bezeichnet man ein Programm, mit dem es möglich ist, Texte einzugeben und zu verändern. Einen der einfachsten Editoren stellt das CLI-Fenster dar, in dem immer nur eine Zeile editiert (verändert) werden kann. Der Komfort wächst mit den Möglichkeiten, die ein Editor bietet. So ist die Shell schon mit mehr Funktionen ausgerüstet. Ähnlich wie der Editor einer Textverarbeitung ist das Notepad aufgebaut. Hier lassen sich Texte mit

verschiedenen Absätzen gestalten und verändern.

Festplatte

Festplatten sind genauso wie die → Floppies Geräte zum Speichern von Daten, die auch noch nach dem Abschalten unseres Amiga erhalten bleiben sollten. Am Anfang dieser Geräte stand das Kassettenlaufwerk, mit dem auf handelsübliche Kassetten Programme und deren Daten aufgezeichnet werden konnten.

Eine Verbesserung brachten dann die Floppies oder Diskettenlaufwerke, die wesentlich schneller und zuverlässiger arbeiteten. Sie waren zwar zu Anfang teurer, sind jetzt aber auf einem sehr guten Preis/Leistungs-Verhältnis angelangt. Ein großer Vorteil liegt bei den Disketten darin, daß es ein Inhaltsverzeichnis gibt und man nicht spulen muß, wenn man ein bestimmtes Programm laden möchte, wie es bei den Kassetten der Fall war.

Eine Festplatte ist nun eine weitere Verbesserung der Disketten. Auch eine Festplatte besteht aus magnetbeschichteten Scheiben. Allerdings sind dies Scheiben aus festem Material und in einer größeren Zahl vorhanden, als es Disketten sind. Auf jede dieser Scheiben können trotz einer etwa gleichen Größe mehr Daten aufgezeichnet werden. Das liegt größtenteils daran, daß eine Festplatte Vakuumversiegelt ist. Man schützt damit den Speicher vor Staub und unreiner Luft. Auch die Zugriffszeit im Vergleich zur Diskette ist bei Festplatten wesentlich niedriger.

File

Als ein File bezeichnet man alle auf einem Datenträger abgespeicherten Benutzer-Daten. Dazu zählen sowohl Programme als auch Da-

ten, die von diesen Programmen erzeugt oder benötigt werden.

Floppy

Floppy ist die umgangssprachliche Bezeichnung für das Diskettenlaufwerk. Man verwendet diesen Begriff für Laufwerk und Disketten. Es handelt sich dabei um ein Gerät zur Speicherung von Daten auf Disketten. Dieses Verfahren wurde gewählt, weil durch die rotierenden Scheiben ein im Verhältnis zur Bandspeicherung sehr schneller Zugriff möglich ist.

Font

siehe → Zeichensatz

Gadget

Gadgets sind von → Intuition verwaltete Klick-Bereiche, die beim Anklicken durch die Maus eine bestimmte Aktion auslösen. So werden die Gadgets z.B. für die → Icons benutzt, um deren Aktivierung zu registrieren. Neben den einfachen Boolean-Gadgets, die nur einen Wahrheitswert produzieren, nämlich, daß sie angeklickt wurden, gibt es noch die String-, die Proportional- und die System-Gadgets. Das String-Gadget wird zur Eingabe von Texten benutzt und ist über die Workbench-Funktion RENAME zu erreichen. Das Proportional-Gadget befindet sich in jedem Verzeichnis-Window der Workbench zum Einstellen des Ausschnitts bei größeren Verzeichnissen. Und die System-Gadgets befinden sich im Window-Rand bzw. in der oberen Zeile der → Screens und werden für die Grund-Funktionen wie Tiefe, Position verändern oder Größe einstellen verwendet.

Guru-Meditation

Beim Amiga kommt es wie bei jedem anderen Computer auch vor, daß innerhalb eines Programms Fehler auftreten. Das wird z.B. bei BASIC mit einer Fehlermeldung abgefangen

und dem Benutzer mitgeteilt. Programmiert man allerdings das → Betriebssystem eines Computers, gibt es diese Auffangvorrichtung nicht und der Computer reagiert gestört. Das zeigt sich meistens dadurch, daß der Bildschirm wild flimmert oder der Amiga gar nicht mehr z.B. auf Tastendrücke reagiert. Man bezeichnet das als Absturz. Damit aber der Programmierer weiß, weswegen der Amiga so reagiert hat und wo etwa der Fehler aufgetreten ist, gibt es die Guru-Meditation. Das ist eine interne Fehlermeldung, bei der im oberen Bereich des Bildschirms ein rot blinkender Kasten erscheint, der → Alert. In ihm findet der Programmierer die Nachricht, die er für die Fehlersuche braucht.

Hardware

Als Hardware bezeichnet man alle festen Dinge, die man anfassen kann. Der Amiga 500 ist Hardware, die → Maus, die → Tastatur, ein zusätzliches Diskettenlaufwerk, ein Drucker und auch die Disketten sind Hardware. Nicht Hardware, sondern → Software sind die Programme auf den Disketten. Die gleiche Unterscheidung macht man auch bei → ROM. Das ROM an sich, also der Chip, ist Hardware. Der Inhalt des ROM - z.B. das Kickstart - ist aber Software.

Icon

Die auf der → Workbench verwendeten Symbole für Diskette, Programme, Daten-Files oder Verzeichnisse nennt man Icons. Jedes der Icons kann über die → Maus angeklickt und somit zum Auslösen einer Funktion benutzt werden.

IFF

Damit Programme von verschiedenen Herstellern die Daten abspeichern und diese untereinander austauschen können, wurde ein Format geschaffen, das sich das Internationale

File Format nennt. Es ermöglicht den Datenaustausch auf den Gebieten Musik, Text und Grafik. Gerade der letzte Bereich wird von allen Grafik-Programmen genutzt. Für Sie als Benutzer ist es Dank dieses Formats möglich, Grafiken von allen Zeichenprogrammen mit ihrem eigenen zu bearbeiten.

Inhaltsverzeichnis

siehe → Directory

Interpreter

Ein Interpreter ist ein Programm, das im Gegensatz zum → Compiler zwar auch einen Programmtext z.B. BASIC übersetzt, diesen aber nicht im Ganzen bearbeitet, sondern immer nur den Befehl, der gerade ausgeführt werden soll. Aus diesem Grund sind Interpreter einfacher zu bedienen, weil der Übersetzungs-Vorgang nicht erst aufgerufen werden muß. Andererseits arbeiten Interpreter auch langsamer, da sie besonders bei Schleifen alle Befehle immer wieder übersetzen müssen. Hier liegt der Vorteil des Compilers begründet, der alles nur einmal übersetzt.

Intuition

Intuition ist der Name der Benutzeroberfläche auf dem Amiga. Sie verwaltet die → Windows, → Gadgets, → Menüs und → Screens, die zur optischen Gliederung und leicht verständlichen Eingabe benutzt werden.

Joystick

Der Joystick ist ein Eingabegerät, das für Spiele verwendet wird. Das ist meist ein Hebel, der sich in die vier Himmelsrichtungen bewegen läßt. Synchron dazu bewegt sich auf dem Bildschirm das Spielobjekt. Zum Joystick gehört auch der Feuerknopf, der auch als → Button bezeichnet wird.

Kickstart

Der Amiga benötigt wie jeder andere Computer auch ein Programm, das ihn zum Leben

erweckt und die Grundaufgaben, die schon in der Einschaltphase auftreten, ausführen kann. Dieses Programm wird als Kickstart bezeichnet (in der Anlehnung an das Beschleunigen eines Automatik-Autos über das Durchtreten des Gaspedals). Es mußte beim ersten Amiga-Modell, dem Amiga 1000, von Diskette nachgeladen werden, weil man dadurch die Veränderung am Kickstart möglichst einfach gestalten wollte. Um aber einen kostengünstigeren Computer zu produzieren, wurde das Kickstart auf ROM gebrannt und ist nun ohne Laden vorhanden.

Library

Wegen der Fülle der Unterroutinen, die im → Betriebssystem untergebracht wurden, hat man sich überlegt, wie diese zu sortieren wären. Als Gliederung sind Libraries (Bibliotheken) herausgekommen, in denen immer nur Funktionen eines Aufgabengebietes enthalten sind. Es gibt Libraries für die verschiedensten Aufgaben: für die Grafikausgabe, für die Diskettenbefehle, für Verwaltungsarbeiten und für Rechenoperationen.

Maschinensprache

siehe → Assembler

Maus

Die Maus ist ein neuartiges Eingabegerät, das es erst seit kurzem gibt. Auch der Amiga besitzt eine Maus, schließlich handelt es sich um einen neuartigen Computer. Wir benutzen die Maus, um damit einfache Eingaben in der → Benutzeroberfläche → Intuition zu tätigen. Mit der rechten und linken Maustaste werden verschiedene Aktionen ausgelöst, die den Ablauf eines Programms durch Eingabeimpulse beeinflussen können.

Die Maus ist neben der → Tastatur das wichtigste Eingabegerät auf dem Amiga. Mit ihr

wird auf dem Bildschirm der Pfeil bewegt, ohne den sich die meisten Eingaben nicht durchführen ließen. Mit der Maus wird die sogenannte virtuelle Eingabe unterstützt, die ihren Schwerpunkt auf der Optik hat und nicht auf dem Eingabegerät. Dies sieht man auch daran, daß für die Bedienung der Maus das Augenmerk auf dem Bildschirm bleiben kann, während man bei der Tastatur oft auf diese blicken muß.

Menü

Der Begriff Menü kommt aus der Gastronomie! Dort bezeichnet man damit die Zusammenstellung von verschiedenen Gerichten eines Mahles. Auf dem Computer werden die verschiedenen Gerichte durch Funktionen ersetzt und das Mahl durch ein Programm. Schon haben wir die Auswahl von Bedienfunktionen innerhalb eines Programms. Die Menüs, verwaltet von → Intuition, werden durch einen Druck auf die rechte Maustaste aktiviert. Dann kann der Benutzer über die Titelleiste der → Screen die groben Kategorien auswählen, in denen er dann verschiedene Funktionen und Einstellungen des Programms wiederfindet.

Multitasking

Ältere Computer, wie z.B. der C64 oder auch die weit verbreiteten PC, lassen es nur zu, daß ein Programm zur Zeit abgearbeitet wird. Der Amiga unterstützt nun durch eine geschickte Software-Organisation, daß ein → Prozessor, der 68000er, mehrere Programme scheinbar gleichzeitig abarbeitet.

Norm

Unter einer Norm versteht man eine Absprache, bei der man sich auf bestimmte Gegebenheiten festlegt, so daß jeder weiß, wie z.B. eine Tastatur angeordnet sein muß. In Deutschland gibt es die DIN (Deutsche Indu-

strie Norm), nach der sehr viele Dinge des persönlichen Lebens festgelegt sind. So legt diese Norm unter anderem fest, wie ein Briefkopf aufgebaut sein soll und welchen Abstand die beiden Kontakte eines Steckers für die Stromversorgung haben müssen.

Auch auf dem Computer gibt es Normen: die Größe der Disketten, das Bildsignal, damit es von jedem Monitor dargestellt werden kann und vieles mehr. Allerdings sind doch einige Dinge den Firmen überlassen, die Computer produzieren. Man hat sich deshalb auf sogenannte → Standards geeinigt, um es dem Anwender leichter zu machen.

Patch

Programme sind so, wie man sie geliefert bekommt, nicht vollkommen und auch leider nicht immer fehlerfrei. Man versucht dann in einigen Fällen, diese Programme durch leichte Korrekturen zu verbessern. So gibt es Veränderungs-Programme, die es erlauben, das AmigaBASIC-Ausgabe-Fenster auf seine volle Größe zu ziehen. Man bezeichnet diese Veränderung als Patch und die Tätigkeit als Patchen.

Es wird im Unterschied zum Programmieren ohne den Programmtext vorgenommen, denn der ist in den meisten Fällen von Orginal-Software wie dem AmigaBASIC nicht bekannt. Der wichtigste Befehl eines Patchers ist POKE und das wichtigste Programm ein Monitor, mit dem der Code des zu patchenden Programms gesichtet werden kann.

Programmiersprache

Als Programmiersprache bezeichnet man eine Sammlung von Anweisungen, die den Computer veranlassen, eine Aufgabe auszuführen. Dafür verwendet man Befehlsworte, die mehr

oder weniger der englischen Sprache angelehnt sind. Je eher die Wörter der Sprache ähneln, desto komplexer sind die Befehle und desto weiter ist man von den eigentlichen Befehlen, die der Prozessor versteht, entfernt. Dann wird ein Hilfsprogramm benötigt, das diese menschenverständlichen Befehle in Chip-verständliche Befehle umsetzt. Dies kann einerseits ein → Interpreter oder auch ein → Compiler sein. Als Hochsprachen der Programmierung sind auf dem Amiga hauptsächlich → BASIC und → C verbreitet. Sehr nahe an den 68000-Prozessor angelehnt ist die → Maschinensprache.

Prozessor

Der Amiga wird von mehreren integrierten Schaltkreisen gesteuert. Einen solchen Schaltkreis nennt man einfach Chip. Da bei mehreren Chips einer das Sagen haben muß, damit alles ordentlich koordiniert werden kann, gibt es immer einen Prozessor, der die Hauptarbeit übernimmt. Im Normalfall ist das der 68000er, der von Commodore eingesetzt wird. Er kann aber auch durch andere Prozessoren, die wesentlich leistungsfähiger sind, ersetzt werden.

RAM

Neben dem → ROM stellt das RAM die zweite Speicherklasse dar. Die Buchstaben stehen für die Abkürzung RANDOM ACCESS MEMORY und bedeuten auf Deutsch wörtlich wahlfreier Zugriff, worunter zu verstehen ist, daß man beliebig lesend oder schreibend auf den Speicher zugreifen kann. Das macht es erst möglich, mit dem Computer zu arbeiten, weil wir Daten lesen und schreiben können.

Der Amiga besitzt nun leider zwei verschiedene Typen von RAM. Einmal das CHIP-RAM, auf das die sogenannten → Custom-

Chips zugreifen können. Das sind die unteren 512 KByte, die leider deswegen auch etwas langsamer im Zugriff sind. Als zweites gibt es das FAST-RAM, das - wie sein Name schon sagt - schneller als das CHIP-RAM ist, weil die Custom-Chips darauf keinen Zugriff haben, der Zeit verschwenden könnte.

RAM-Disk

Neben den normalen → Disketten wird auf dem Amiga noch eine sogenannte RAM-Disk simuliert. Das ist eine nur im Speicher existierende Diskette, auf der Daten und Programme in gleicher Weise - aber natürlich schneller - abgespeichert und gelesen werden können. Allerdings ist der Inhalt nach dem Löschen des Speichers (sprich Ausschalten oder → Reset) gelöscht und verloren.

Requester

Der Requester ist eine Nachfrage des Programms an den Benutzer. Dafür wird ein deutlich gekennzeichneter Bildschirm-Bereich verwendet, in dem neben einem Informationstext noch → Gadgets zur Auswahl stehen, die die gestellte Nachfrage beantworten. Sie kennen diesen Vorgang vielleicht von der Workbench, bei der die sogenannten System-Requester erscheinen. Diese Anfragen kommen vom System und enthalten meistens Anweisungen, die Diskette einzulegen oder einen wichtigen Vorgang zu bestätigen. Das ist z.B. beim Löschen von Daten oder Programmen nötig, da man sich beim ersten Mal versehen haben kann und durch die zweifache Bestätigung eine gewisse Sicherheit vor Fehlaufrufen solcher gefährlichen Funktionen besteht.

Reset

Der Reset, der auch als Warmstart bezeichnet wird, ist ähnlich dem Ausschalten. Der Computer springt in einen Teil seines Betriebssystems, in dem der Speicher gelöscht und das

ganze System neu eingerichtet wird. Beim Amiga löst man dies durch die Tastenkombination [Ctrl]-[Amiga]-[Amiga] aus. Alle Tasten müssen etwa eine Sekunde gleichzeitig niedergedrückt sein.

ROM

Der Begriff ROM steht für Read Only Memory und bedeutet übersetzt auf Deutsch "Nur Lese Speicher", was heißen soll, daß man auf diesem Speicher zwar Daten auslesen kann, in diesen aber keine neuen Daten geschrieben werden können, wie es beim → RAM der Fall ist. Der Vorteil dieses Speichers liegt darin, daß sein Inhalt auch beim Ausschalten des Computers erhalten bleibt und so nicht wieder hineingeschrieben werden muß. Der Nachteil liegt aber in der Unveränderbarkeit des Speichers.

Screen

Mit Screen bezeichnet man im Englischen ganz einfach den Bildschirm. Dies ist auch bei anderen Computern so üblich. Beim Amiga hat das Wort eine besondere Bedeutung, weil es nicht den faßbaren Bildschirm bezeichnet, sondern die Fläche, auf der Grafik und Text dargestellt werden. Dies macht es auch möglich, über → Intuition mehrere Screens darzustellen, die getrennt verwaltet werden. Auf dem "echten" Bildschirm kann natürlich immer nur eine Screen ganz zu sehen sein. Es können sich aber mehrere Screens einen Bildschirm teilen. Oder, was durch Hardware-Erweiterungen unterstützt wird, man stellt verschiedene Screens auf verschiedenen Bildschirmen dar.

Software

Als Software bezeichnet man alle Programme, die auf einem Computer laufen. Software (weiche Ware) soll ausdrücken, daß der Inhalt mehr auf gedankliche Arbeit zurückzuführen

ist. Beispiele für Software sind alle Programme, die auf dem Amiga laufen wie die Clock, das Notepad, der Calculator, die Demos usw. Auch das Amiga-Betriebssystem ist Software, obwohl es sich als Kickstart im ROM befindet!

Standard

Standards sind in der Computer-Branche nicht mehr wegzudenken. Unter einem Standard versteht man die Festsetzung einer Richtlinie, nach der etwas zu geschehen hat. So gibt es einen Datenübertragungs-Standard, damit man auch Drucker von Fremdherstellern nehmen kann und nicht den von Commodore kaufen muß. Es gibt einen Befehls-Standard von Microsoft für BASIC-Befehle. Damit bezeichnet man die Mindest-Befehle, die ein BASIC-Interpreter anbieten muß. Auch die Parameter, die dahinter folgen, sind festgelegt. Nur so ist es möglich, daß man nicht nur auf einem Computer arbeitet, sondern sein Wissen für viele andere Modelle ausnutzen kann.

Ohne Standards wäre es praktisch unmöglich, Daten zwischen zwei Computern auszutauschen. Aber, wie Sie wissen, haben wir hinten an unserem Amiga 500 eine parallele und eine serielle Schnittstelle zum Datenaustausch. Auch dies ist ein Standard, nämlich der Centronics- oder der RS-232-Standard.

Tastatur

Mit Tastatur oder auch Keyboard bezeichnet man das Eingabegerät des Amiga, mit dem Zahlen, Zeichen und Buchstaben eingegeben werden können. Die Tastatur des Amiga umfaßt 10 Funktionstasten, 20 Zifferntasten (die doppelte Ausführung wurde verwendet, damit auch über den Ziffernblock auf der rechten Seite Zahlen eingegeben werden können), 29 alphanumerische Tasten und 31 Sondertasten.

Abgeschlossen wird diese Aufzählung mit 4 Cursortasten, die immer die richtige Richtung angeben.

Interessant ist die Möglichkeit, mit der Tastatur auch die → Maus zu simulieren. Dafür muß man nur die rechte [Amiga]-Taste gedrückt halten und gleichzeitig die Cursortasten betätigen.

Verzeichnis

siehe → Directory

Virus

Mit Virus bezeichnet man, angelehnt an die Biologie, ein Programm, das es sich zum Ziel gesetzt hat, sich zu vermehren - und das auch auf Kosten anderer. In diesem Fall sind andere Programme und Daten im Computer oder auf Diskette betroffen. Ein Virus ist teilweise so zerstörerisch, daß es den Computer, unseren Amiga, abstürzen läßt, so daß kein Programm mehr läuft, oder daß eine Diskette gelöscht bzw. mit anderen Daten beschrieben wird. Der Benutzer kann sich nur durch Sicherung seiner Daten und Programme und durch ständige Kontrollen vor Viren schützen. Ist der Amiga erst einmal befallen, hilft nur das Ausschalten und das Löschen der Disketten, auf denen sich das Virus-Programm befindet.

Window

Der Grafikbereich eines Bildschirms wird im Groben durch → Screens organisiert. Ein Window ist eine Fläche darauf, in der Text oder Grafik dargestellt werden kann. Im allgemeinen sind Windows über einen Rand deutlich von den anderen Windows abgetrennt. Sie ermöglichen erst die Ausgabe im Multitasking, denn durch die Trennung kann jedes Programm in seinem eigenen Window Daten, Texte oder Zeichnungen ausgeben, ohne mit den anderen zu kollidieren.

Workbench

Die Arbeitsoberfläche, mit der wir fast täglich arbeiten, nennt man Workbench - angelehnt an das englische Wort für Schreibtisch. Mit der Workbench lassen sich alle wichtigen Disketten-Operationen durchführen. Außerdem können von ihr Programme gestartet werden. Das besondere an der Workbench ist aber, daß sie symbol-orientiert arbeitet und so auch für den Anfänger leicht verständlich ist. Es entfällt eine lange Einarbeitungszeit.

Zeichensatz

Der Amiga gibt Texte in einer bestimmten Schrift auf dem Bildschirm aus. Man bezeichnet diese Schrift als Zeichensatz oder engl. Font. Da der Amiga sehr vielseitig ist, gibt es verschiedene Zeichensätze. Zwei davon sind fest eingebaut und heißen Topaz 8 und 9. Dabei steht die Zahl für die Anzahl der Grafik-Zeilen, die jede Schrift in Anspruch nimmt. Davon hängt es nämlich ab, wie viele Zeichen in eine Zeile passen. Je mehr, desto besser aber auch desto schwerer lesbar. Aber nicht nur die Anzahl der Zeichen pro Zeile (CPL [Characters per Line]), sondern auch das Aussehen der Schrift wird zur Unterscheidung herangezogen. Es gibt zweckmäßige und verschnörkelte Schriften. Je nach Belieben.

Zeileneditor

Der Zeileneditor ist eine spezielle Gattung des Editors. Man unterscheidet hier zwei Typen. Mit dem Zeileneditor ist es immer nur möglich, eine Zeile des Textes zu bearbeiten. Jede andere muß erst gesondert angewählt werden. In vielen Fällen hat man auch andere Zeilen nicht auf dem Bildschirm. Der Bildschirm- oder FullScreen-Editor hingegen stellt alle Zeilen des zu bearbeitenden Textes, soweit es möglich ist, auf dem Bildschirm dar. Mit Hilfe der Cursortasten kann man dann beliebig auswählen, welche Zeile bearbeitet werden soll.

Anhang C: Funktionen der Amiga-Libraries

In diesem Anhang finden Sie ausgewählte Libraries des Amiga-Betriebssystems. Zu jeder Library gibt es eine Tabelle mit all ihren Funktionen. Diese Tabellen sollen von Ihnen für die Programmier-Arbeit in BASIC, Assembler und C genutzt werden können.

Die Tabellen sind folgendermaßen aufgebaut: In den ersten beiden Spalten sind die Offsets der Funktionen zu finden. Da wir diese nicht von BASIC aus benötigen, sind die Zahlen in der ersten Spalte hexadezimal für den C-Programmierer und in der zweiten Spalte dezimal für den Assembler-Programmierer. Weiterhin steht in jeder Zeile der Funktionsname mit den benötigten Parametern. Diese sind hier mit ihren Namen aufgeführt, damit man schnell erkennen kann, welche Werte die Funktion fordert. Außerdem gibt es dazu - für den Assembler-Programmierer - noch die Register, in denen die Werte übergeben werden. Die Reihenfolge ist dabei analog zu den anderen Parametern.

intuition.library

-0x001E	- 30	OpenIntuition()()
-0x0024	- 36	Intuition(Ievent)(A0)
-0x002A	- 42	AddGadget(AddPtr, Gadget, Position)(A0/A1,D0)
-0x0030	- 48	ClearDMRequest(Window)(A0)
-0x0036	- 54	ClearMenuStrip(Window)(A0)
-0x003C	- 60	ClearPointer(Window)(A0)
-0x0042	- 66	CloseScreen(Screen)(A0)
-0x0048	- 72	CloseWindow(Window)(A0)
-0x004E	- 78	CloseWorkBench()()
-0x0054	- 84	CurrentTime(Seconds, Micros)(A0/A1)
-0x005A	- 90	DisplayAlert(AlertNumber, String, Height)(D0/A0,D1)
-0x0060	- 96	DisplayBeep(Screen)(A0)
-0x0066	- 102	DoubleClick(Sseconds, Smicros, Cseconds,Cmicros) (D0/D1/D2/D3)
-0x006C	- 108	DrawBorder(RPort, Border, LeftOffset, TopOffset) (A0/A1,D0/D1)
-0x0072	- 114	DrawImage(RPort, Image, LeftOffset, TopOffset)(A0/A1,D0/D1)
-0x0078	- 120	EndRequest(Requester, Window)(A0/A1)
-0x007E	- 126	GetDefPrefs(Preferences, Size)(A0,D0)
-0x0084	- 132	GetPrefs(Preferences, Size)(A0,D0)
-0x008A	- 138	InitRequester(Req)(A0)

-0x0090 - 144 ItemAddress(MenuStrip, MenuNumber)(A0,D0)
-0x0096 - 150 ModifyIDCMP(Window, Flags)(A0,D0)
-0x009C - 156 ModifyProp(Gadget, Ptr, Req, Flags, HPos, VPos, HBody, VBody)(A0/A1/A2,D0/D1/D2/D3/D4)
-0x00A2 - 162 MoveScreen(Screen, Dx, Dy)(A0,D0/D1)
-0x00A8 - 168 MoveWindow(Window, Dx, Dy)(A0,D0/D1)
-0x00AE - 174 OffGadget(Gadget, Ptr, Req)(A0/A1/A2)
-0x00B4 - 180 OffMenu(Window, MenuNumber)(A0,D0)
-0x00BA - 186 OnGadget(Gadget, Ptr, Req)(A0/A1/A2)
-0x00C0 - 192 OnMenu(Window, MenuNumber)(A0,D0)
-0x00C6 - 198 OpenScreen(OSargs)(A0)
-0x00CC - 204 OpenWindow(OWargs)(A0)
-0x00D2 - 210 OpenWorkBench()()
-0x00D8 - 216 PrintIText(Rp, Itext, Left, Top)(A0/A1,D0/D1)
-0x00DE - 222 RefreshGadgets(Gadgets, Ptr, Req)(A0/A1/A2)
-0x00E4 - 228 RemoveGadget(RemPtr, Gadget)(A0/A1)
-0x00EA - 234 ReportMouse(Boolean, Window)(D0/A0)
-0x00F0 - 240 Request(Requester, Window)(A0/A1)
-0x00F6 - 246 ScreenToBack(Screen)(A0)
-0x00FC - 252 ScreenToFront(Screen)(A0)
-0x0102 - 258 SetDMRequest(Window, Req)(A0/A1)
-0x0108 - 264 SetMenuStrip(Window, Menu)(A0/A1)
-0x010E - 270 SetPointer(Window, Pointer, Height, Width, Xoffset, Yoffset) (A0/A1,D0/D1/D2/D3)
-0x0114 - 276 SetWindowTitles(Window, Windowtitle, Screentitle) (A0/A1/A2)
-0x011A - 282 ShowTitle(Screen, ShowIt)(A0,D0)
-0x0120 - 288 SizeWindow(Window, Dx, Dy)(A0,D0/D1)
-0x0126 - 294 ViewAddress()()
-0x012C - 300 ViewPortAddress(Window)(A0)
-0x0132 - 306 WindowToBack(Window)(A0)
-0x0138 - 312 WindowToFront(Window)(A0)
-0x013E - 318 WindowLimits(Window, Minwidth, Minheight, Maxwidth, Maxheight)(A0,D0/D1/D2/D3)
-0x0144 - 324 SetPrefs(Preferences, Size, Flag)(A0,D0/D1)
-0x014A - 330 IntuiTextLength(Itext)(A0)
-0x0150 - 336 WBenchToBack()()
-0x0156 - 342 WBenchToFront()()
-0x015C - 348 AutoRequest(Window, Body, PText, NText, PFlag, NFlag, W, H)(A0,A1,A2,A3,D0,D1,D2,D3)
-0x0162 - 354 BeginRefresh(Window)(A0)
-0x0168 - 360 BuildSysRequest(Window, Body, PosText, NegText, Flags, W, H)(A0,A1,A2,A3,D0,D1,D2)
-0x016E - 366 EndRefresh(Window, Complete)(A0,D0)
-0x0174 - 372 FreeSysRequest(Window)(A0)
-0x017A - 378 MakeScreen(Screen)(A0)
-0x0180 - 384 RemakeDisplay()()
-0x0186 - 390 RethinkDisplay()()

-0x018C - 396 AllocRemember(RememberKey, Size, Flags)(A0,D0,D1)
-0x0192 - 402 AlohaWorkbench(Wbport)(A0)
-0x0198 - 408 FreeRemember(RememberKey, ReallyForget)(A0,D0)
-0x019E - 414 LockIBase(Dontknow)(D0)
-0x01A4 - 420 UnlockIBase(IBLock)(A0)
-0x01AA - 426 GetScreenData(Buffer, Size, Type, Screen)(A0,D0, D1,A1)
-0x01B0 - 432 RefreshGList(Gadgets, Ptr, Req, NumGad)(A0/A1/A2, D0)
-0x01B6 - 438 AddGList(AddPtr, Gadget, Position, NumGad, Requester) (A0/A1,D0/D1/A2)
-0x01BC - 444 RemoveGList(RemPtr, Gadget, NumGad)(A0/ A1,D0)
-0x01C2 - 450 ActivateWindow(Window)(A0)
-0x01C8 - 456 RefreshWindowFrame(Window)(A0)
-0x01CE - 462 ActivateGadget(Gadgets, Window, Req)(A0/ A1/A2)
-0x01D4 - 468 NewModifyProp(Gadget, Ptr, Req, Flags, HPos, VPos, HBody, VBody, NumGad)(A0/A1/A2,D0/D1/D2/D3/D4/D5)

graphics.library

-0x001E - 30 BltBitMap(SrcBitMap, SrcX, SrcY, DestBitMap, DestX, DestY, SizeX, SizeY, Minterm, Mask, TempA)(A0,D0 /D1,A1,D2/D3/D4/D5/D6/D7/A2)
-0x0024 - 36 BltTemplate(Source, SrcX, SrcMod, DestRastPort, DestX, DestY, SizeX, SizeY)(A0,D0/D1/A1,D2/D3/ D4/D5)
-0x002A - 42 ClearEOL(RastPort)(A1)
-0x0030 - 48 ClearScreen(RastPort)(A1)
-0x0036 - 54 TextLength(RastPort, String, Count)(A1,A0,D0)
-0x003C - 60 Text(RastPort, String, Count)(A1,A0,D0)
-0x0042 - 66 SetFont(RastPortID, TextFont)(A1,A0)
-0x0048 - 72 OpenFont(TextAttr)(A0)
-0x004E - 78 CloseFont(TextFont)(A1)
-0x0054 - 84 AskSoftStyle(RastPort)(A1)
-0x005A - 90 SetSoftStyle(RastPort, Style, Enable)(A1,D0/D1)
-0x0060 - 96 AddBob(Bob, RastPort)(A0,A1)
-0x0066 - 102 AddVSprite(VSprite, RastPort)(A0/A1)
-0x006C - 108 DoCollision(RasPort)(A1)
-0x0072 - 114 DrawGList(RastPort, ViewPort)(A1,A0)
-0x0078 - 120 InitGels(DummyHead, DummyTail, GelsInfo) (A0/A1/A2)
-0x007E - 126 InitMasks(VSprite)(A0)
-0x0084 - 132 RemIBob(Bob, RastPort, ViewPort)(A0/A1/A2)
-0x008A - 138 RemVSprite(VSprite)(A0)
-0x0090 - 144 SetCollision(Type, Routine, GelsInfo)(D0/A0/A1)
-0x0096 - 150 SortGList(RastPort)(A1)
-0x009C - 156 AddAnimOb(Obj, AnimationKey, RastPort)(A0/A1/A2)
-0x00A2 - 162 Animate(AnimationKey, RastPort)(A0/A1)
-0x00A8 - 168 GetGBuffers(AnimationObj, RastPort, DoubleBuffer) (A0/A1,D0)
-0x00AE - 174 InitGMasks(AnimationObj)(A0)

-0x00B4 - 180 DrawEllipse(RastPort, Cx, Cy, A, B)(A1,D0/D1/D2/D3)
-0x00BA - 186 AreaEllipse(RastPort, Cx, Cy, A, B)(A1,D0/D1/D2/D3)
-0x00C0 - 192 LoadRGB4(ViewPort, Colors, Count)(A0/A1,D0)
-0x00C6 - 198 InitRastPort(RastPort)(A1)
-0x00CC - 204 InitVPort(ViewPort)(A0)
-0x00D2 - 210 MrgCop(View)(A1)
-0x00D8 - 216 MakeVPort(View, ViewPort)(A0/A1)
-0x00DE - 222 LoadView(View)(A1)
-0x00E4 - 228 WaitBlit()
-0x00EA - 234 SetRast(RastPort, Color)(A1,D0)
-0x00F0 - 240 Move(RastPort, X, Y)(A1,D0/D1)
-0x00F6 - 246 Draw(RastPort, X, Y)(A1,D0/D1)
-0x00FC - 252 AreaMove(RastPort, X, Y)(A1,D0/D1)
-0x0102 - 258 AreaDraw(RastPort, X, Y)(A1,D0/D1)
-0x0108 - 264 AreaEnd(RastPort)(A1)
-0x010E - 270 WaitTOF()
-0x0114 - 276 QBlit(Blit)(A1)
-0x011A - 282 InitArea(AreaInfo, VectorTable, VectorTableSize)(A0/A1,D0)
-0x0120 - 288 SetRGB4(ViewPort, Index, R, G, B)(A0,D0/D1/D2/D3)
-0x0126 - 294 QBSBlit(Blit)(A1)
-0x012C - 300 BltClear(Memory, Size, Flags)(A1,D0/D1)
-0x0132 - 306 RectFill(RastPort, Xl, Yl, Xu, Yu)(A1,D0/D1/D2/D3)
-0x0138 - 312 BltPattern(RastPort, Ras, Xl, Yl, MaxX, MaxY,FillBytes)
(a1,a0,D0/D1/D2/D3/D4)
-0x013E - 318 ReadPixel(RastPort, X, Y)(A1,D0/D1)
-0x0144 - 324 WritePixel(RastPort, X, Y)(A1,D0/D1)
-0x014A - 330 Flood(RastPort, Mode, X, Y)(A1,D2,D0/D1)
-0x0150 - 336 PolyDraw(RastPort, Count, PolyTable)(A1,D0,A0)
-0x0156 - 342 SetAPen(RastPort, Pen)(A1,D0)
-0x015C - 348 SetBPen(RastPort, Pen)(A1,D0)
-0x0162 - 354 SetDrMd(RastPort, DrawMode)(A1,D0)
-0x0168 - 360 InitView(View)(A1)
-0x016E - 366 CBump(CopperList)(A1)
-0x0174 - 372 CMove(CopperList, Destination, Data)(A1,D0/D1)
-0x017A - 378 CWait(CopperList, X, Y)(A1,D0/D1)
-0x0180 - 384 VBeamPos()
-0x0186 - 390 InitBitMap(BitMap, Depth, Width, Height)(A0,D0/D1/D2)
-0x018C - 396 ScrollRaster(RastPort, DX, DY, Minx, Miny, Maxx, Maxy)
(A1,D0/D1/D2/D3/D4/D5)
-0x0192 - 402 WaitBOVP(Viewport)(a0)
-0x0198 - 408 GetSprite(Simplesprite, Num)(a0,d0)
-0x019E - 414 FreeSprite(Num)(d0)
-0x01A4 - 420 ChangeSprite(Vp, Simplesprite, Data)(a0/a1/a2)
-0x01AA - 426 MoveSprite(Viewport, Simplesprite, X, Y)(a0/a1,d0/d1)
-0x01B0 - 432 LockLayerRom(Layer)(a5)
-0x01B6 - 438 UnlockLayerRom(Layer)(a5)
-0x01BC - 444 SyncSBitMap(L)(a0)
-0x01C2 - 450 CopySBitMap(L)(a0)

-0x01C8 - 456 OwnBlitter()()
-0x01CE - 462 DisownBlitter()()
-0x01D4 - 468 InitTmpRas(Tmpras, Buff, Size)(a0/a1,d0)
-0x01DA - 474 AskFont(RastPort, TextAttr)(A1,A0)
-0x01E0 - 480 AddFont(TextFont)(A1)
-0x01E6 - 486 RemFont(TextFont)(A1)
-0x01EC - 492 AllocRaster(Width, Height)(D0/D1)
-0x01F2 - 498 FreeRaster(Planeptr, Width, Height)(A0,D0/D1)
-0x01F8 - 504 AndRectRegion(Rgn, Rect)(A0/A1)
-0x01FE - 510 OrRectRegion(Rgn, Rect)(A0/A1)
-0x0204 - 516 NewRegion()()
-0x020A - 522 ClearRectRegion(Rgn)(A0/A1)
-0x0210 - 528 ClearRegion(Rgn)(A0)
-0x0216 - 534 DisposeRegion(Rgn)(A0)
-0x021C - 540 FreeVPortCopLists(Viewport)(a0)
-0x0222 - 546 FreeCopList(Coplist)(a0)
-0x0228 - 552 ClipBlit(Srcrp, SrcX, SrcY, Destrp, DestX, DestY,SizeX, SizeY, Minterm)(A0,D0/D1,A1,D2/D3/D4/D5/D6)
-0x022E - 558 XorRectRegion(Rgn, Rect)(a0/a1)
-0x0234 - 564 FreeCprList(Cprlist)(a0)
-0x023A - 570 GetColorMap(Entries)(d0)
-0x0240 - 576 FreeColorMap(Colormap)(a0)
-0x0246 - 582 GetRGB4(Colormap, Entry)(a0,d0)
-0x024C - 588 ScrollVPort(Vp)(a0)
-0x0252 - 594 UCopperListInit(Copperlist, Num)(a0,d0)
-0x0258 - 600 FreeGBuffers(AnimationObj, RastPort, DoubleBuffer)(A0/A1,D0)
-0x025E - 606 BltBitMapRastPort(Srcbm, Srcx, Srcy, Destrp, DestX, DestY, SizeX, SizeY, Minterm)(A0,D0/ D1,A1,D2/D3/D4/D5/D6)
-0x0264 - 612 OrRegionRegion(Src, Dst)(a0/a1)
-0x026A - 618 XorRegionRegion(Src, Dst)(a0/a1)
-0x0270 - 624 AndRegionRegion(Src, Dst)(a0/a1)
-0x0276 - 630 SetRGB4CM(Cm, I, R, G, B)(a0,d0/d1/d2/d3)
-0x027C - 636 BltMaskBitMapRastPort(Srcbm, Srcx, Srcy, Destrp, DestX, DestY, SizeX, SizeY, Minterm, Bltmask)(A0, D0/D1,A1,D2/D3/D4/D5/D6,A2)
-0x0282 - 642 GraphicsReserved1()()
-0x0288 - 648 GraphicsReserved2()()
-0x028E - 654 AttemptLockLayerRom(Layer)(a5)

dos.library

-0x001E - 30 Open(Name, AccessMode)(D1/D2)
-0x0024 - 36 Close(File)(D1)
-0x002A - 42 Read(File, Buffer, Length)(D1/D2/D3)
-0x0030 - 48 Write(File, Buffer, Length)(D1/D2/D3)

-0x0036 - 54 Input()
-0x003C - 60 Output()
-0x0042 - 66 Seek(File, Position, Offset)(D1/D2/D3)
-0x0048 - 72 DeleteFile(Name)(D1)
-0x004E - 78 Rename(OldName, NewName)(D1/D2)
-0x0054 - 84 Lock(Name, Type)(D1/D2)
-0x005A - 90 UnLock(Lock)(D1)
-0x0060 - 96 DupLock(Lock)(D1)
-0x0066 - 102 Examine(Lock, FileInfoBlock)(D1/D2)
-0x006C - 108 ExNext(Lock, FileInfoBlock)(D1/D2)
-0x0072 - 114 Info(Lock, ParameterBlock)(D1/D2)
-0x0078 - 120 CreateDir(Name)(D1)
-0x007E - 126 CurrentDir(Lock)(D1)
-0x0084 - 132 IoErr()
-0x008A - 138 CreateProc(Name, Pri, SegList, StackSize)(D1/D2/D3/D4)
-0x0090 - 144 Exit(ReturnCode)(D1)
-0x0096 - 150 LoadSeg(FileName)(D1)
-0x009C - 156 UnLoadSeg(Segment)(D1)
-0x00A2 - 162 GetPacket(Wait)(D1)
-0x00A8 - 168 QueuePacket(Packet)(D1)
-0x00AE - 174 DeviceProc(Name)(D1)
-0x00B4 - 180 SetComment(Name, Comment)(D1/D2)
-0x00BA - 186 SetProtection(Name, Mask)(D1/D2)
-0x00C0 - 192 DateStamp(Date)(D1)
-0x00C6 - 198 Delay(Timeout)(D1)
-0x00CC - 204 WaitForChar(File, Timeout)(D1/D2)
-0x00D2 - 210 ParentDir(Lock)(D1)
-0x00D8 - 216 IsInteractive(File)(D1)
-0x00DE - 222 Execute(String, File, File)(D1/D2/D3)

exec.library

-0x001E - 30 Supervisor()
-0x0024 - 36 ExitIntr()
-0x002A - 42 Schedule()
-0x0030 - 48 Reschedule()
-0x0036 - 54 Switch()
-0x003C - 60 Dispatch()
-0x0042 - 66 Exception()
-0x0048 - 72 InitCode(StartClass, Version)(D0/D1)
-0x004E - 78 InitStruct(InitTable, Memory, Size)(A1/A2,D0)
-0x0054 - 84 MakeLibrary(FuncInit, StructInit, LibInit, DataSize, CodeSize)(A0/A1/A2,D0/D1)
-0x005A - 90 MakeFunctions(Target, FunctionArray, FuncDispBase) (A0,A1,A2)
-0x0060 - 96 FindResident(Name)(A1)
-0x0066 - 102 InitResident(Resident, SegList)(A1,D1)

-0x006C - 108 Alert(AlertNum, Parameters)(D7,A5)
-0x0072 - 114 Debug()
-0x0078 - 120 Disable()
-0x007E - 126 Enable()
-0x0084 - 132 Forbid()
-0x008A - 138 Permit()
-0x0090 - 144 SetSR(NewSR, Mask)(D0/D1)
-0x0096 - 150 SuperState()
-0x009C - 156 UserState(SysStack)(D0)
-0x00A2 - 162 SetIntVector(IntNumber, Interrupt)(D0/A1)
-0x00A8 - 168 AddIntServer(IntNumber, Interrupt)(D0/A1)
-0x00AE - 174 RemIntServer(IntNumber, Interrupt)(D0/A1)
-0x00B4 - 180 Cause(Interrupt)(A1)
-0x00BA - 186 Allocate(FreeList, ByteSize)(A0,D0)
-0x00C0 - 192 Deallocate(FreeList, MemoryBlock, ByteSize)(A0/A1,D0)
-0x00C6 - 198 AllocMem(ByteSize, Requirements)(D0/D1)
-0x00CC - 204 AllocAbs(ByteSize, Location)(D0/A1)
-0x00D2 - 210 FreeMem(MemoryBlock, ByteSize)(A1,D0)
-0x00D8 - 216 AvailMem(Requirements)(D1)
-0x00DE - 222 AllocEntry(Entry)(A0)
-0x00E4 - 228 FreeEntry(Entry)(A0)
-0x00EA - 234 Insert(List, Node, Pred)(A0/A1/A2)
-0x00F0 - 240 AddHead(List, Node)(A0/A1)
-0x00F6 - 246 AddTail(List, Node)(A0/A1)
-0x00FC - 252 Remove(Node)(A1)
-0x0102 - 258 RemHead(List)(A0)
-0x0108 - 264 RemTail(List)(A0)
-0x010E - 270 Enqueue(List, Node)(A0/A1)
-0x0114 - 276 FindName(List, Name)(A0/A1)
-0x011A - 282 AddTask(Task, InitPC, FinalPC)(A1/A2/A3)
-0x0120 - 288 RemTask(Task)(A1)
-0x0126 - 294 FindTask(Name)(A1)
-0x012C - 300 SetTaskPri(Task, Priority)(A1,D0)
-0x0132 - 306 SetSignal(NewSignals, SignalSet)(D0/D1)
-0x0138 - 312 SetExcept(NewSignals, SignalSet)(D0/D1)
-0x013E - 318 Wait(SignalSet)(D0)
-0x0144 - 324 Signal(Task, SignalSet)(A1,D0)
-0x014A - 330 AllocSignal(SignalNum)(D0)
-0x0150 - 336 FreeSignal(SignalNum)(D0)
-0x0156 - 342 AllocTrap(TrapNum)(D0)
-0x015C - 348 FreeTrap(TrapNum)(D0)
-0x0162 - 354 AddPort(Port)(A1)
-0x0168 - 360 RemPort(Port)(A1)
-0x016E - 366 PutMsg(Port, Message)(A0/A1)
-0x0174 - 372 GetMsg(Port)(A0)
-0x017A - 378 ReplyMsg(Message)(A1)
-0x0180 - 384 WaitPort(Port)(A0)
-0x0186 - 390 FindPort(Name)(A1)

-0x018C - 396 AddLibrary(Library)(A1)
-0x0192 - 402 RemLibrary(Library)(A1)
-0x0198 - 408 OldOpenLibrary(LibName)(A1)
-0x019E - 414 CloseLibrary(Library)(A1)
-0x01A4 - 420 SetFunction(Library, FuncOffset, FuncEntry)(A1,A0,D0)
-0x01AA - 426 SumLibrary(Library)(A1)
-0x01B0 - 432 AddDevice(Device)(A1)
-0x01B6 - 438 RemDevice(Device)(A1)
-0x01BC - 444 OpenDevice(DevName, Unit, IoRequest, Flags)
(A0,D0/A1,D1)
-0x01C2 - 450 CloseDevice(IoRequest)(A1)
-0x01C8 - 456 DoIO(IoRequest)(A1)
-0x01CE - 462 SendIO(IoRequest)(A1)
-0x01D4 - 468 CheckIO(IoRequest)(A1)
-0x01DA - 474 WaitIO(IoRequest)(A1)
-0x01E0 - 480 AbortIO(IoRequest)(A1)
-0x01E6 - 486 AddResource(Resource)(A1)
-0x01EC - 492 RemResource(Resource)(A1)
-0x01F2 - 498 OpenResource(ResName, Version)(A1,D0)
-0x01F8 - 504 RawIOInit()
-0x01FE - 510 RawMayGetChar()
-0x0204 - 516 RawPutChar(Char)(d0)
-0x020A - 522 RawDoFmt()(A0/A1/A2/A3)
-0x0210 - 528 GetCC()
-0x0216 - 534 TypeOfMem(Address)(A1);
-0x021C - 540 Procure(Semaport, BidMsg)(A0/A1)
-0x0222 - 546 Vacate(Semaport)(A0)
-0x0228 - 552 OpenLibrary(LibName, Version)(A1,D0)
-0x022E - 558 InitSemaphore(SigSem)(A0)
-0x0234 - 564 ObtainSemaphore(SigSem)(A0)
-0x023A - 570 ReleaseSemaphore(SigSem)(A0)
-0x0240 - 576 AttemptSemaphore(SigSem)(A0)
-0x0246 - 582 ObtainSemaphoreList(SigSem)(A0)
-0x024C - 588 ReleaseSemaphoreList(SigSem)(A0)
-0x0252 - 594 FindSemaphore(SigSem)(A1)
-0x0258 - 600 AddSemaphore(SigSem)(A1)
-0x025E - 606 RemSemaphore(SigSem)(A1)
-0x0264 - 612 SumKickData()
-0x026A - 618 AddMemList(Size, Attributes, Pri, Base, Name)
(D0/D1/D2/A0/A1)
-0x0270 - 624 CopyMem(Source, Dest, Size)(A0/A1,D0)
-0x0276 - 630 CopyMemQuick(Source, Dest, Size)(A0/A1,D0)

diskfont.library

-0x001E - 30 OpenDiskFont(TextAttr)(A0)
-0x0024 - 36 AvailFonts(Buffer, BufBytes, Flags)(A0,D0/D1)

-0x002A - 42 NewFontContents(FontsLock, FontName)(A0/A1)
-0x0030 - 48 DisposeFontContents(FontContentsHeader)(A1)

icon.library

-0x001E - 30 GetWBObject(Name)(A0)
-0x0024 - 36 PutWBObject(Name, Object)(A0/A1)
-0x002A - 42 GetIcon(Name, Icon, Freelist)(A0/A1/A2)
-0x0030 - 48 PutIcon(Name, Icon)(A0/A1)
-0x0036 - 54 FreeFreeList(Freelist)(A0)
-0x003C - 60 FreeWBObject(WBObject)(A0)
-0x0042 - 66 AllocWBObject()()
-0x0048 - 72 AddFreeList(Freelist, Mem, Size)(A0/A1/A2)
-0x004E - 78 GetDiskObject(Name)(A0)
-0x0054 - 84 PutDiskObject(Name, Diskobj)(A0,A1)
-0x005A - 90 FreeDiskObject(Diskobj)(A0)
-0x0060 - 96 FindToolType(ToolTypeArray, TypeName)(A0/A1)
-0x0066 - 102 MatchToolValue(TypeString, Value)(A0/A1)
-0x006C - 108 BumpRevision(Newname, Oldname)(A0/A1)

mathffp.library

-0x001E - 30 SPFix(Float)(D0)
-0x0024 - 36 SPFlt(Integer)(D0)
-0x002A - 42 SPCmp(LeftFloat, RightFloat)(D1,D0)
-0x0030 - 48 SPTst(Float)(D1)
-0x0036 - 54 SPAbs(Float)(D0)
-0x003C - 60 SPNeg(Float)(D0)
-0x0042 - 66 SPAdd(LeftFloat, RightFloat)(D1,D0)
-0x0048 - 72 SPSub(LeftFloat, RightFloat)(D1,D0)
-0x004E - 78 SPMul(LeftFloat, RightFloat)(D1,D0)
-0x0054 - 84 SPDiv(LeftFloat, RightFloat)(D1,D0)
-0x005A - 90 SPFloor(Float)(D0)
-0x0060 - 96 SPCeil(Float)(D0)

mathieeedoubbas.library

-0x001E - 30 IEEEDPFix(Double)(D0/D1)
-0x0024 - 36 IEEEDPFlt(Integer)(D0)
-0x002A - 42 IEEEDPCmp(Double, Double)(D0/D1/D2/D3)
-0x0030 - 48 IEEEDPTst(Double)(D0/D1)
-0x0036 - 54 IEEEDPAbs(Double)(D0/D1)
-0x003C - 60 IEEEDPNeg(Double)(D0/D1)
-0x0042 - 66 IEEEDPAdd(Double, Double)(D0/D1/D2/D3)

-0x0048 - 72 IEEEDPSub(Double, Double)(D0/D1/D2/D3)
-0x004E - 78 IEEEDPMul(Double, Double)(D0/D1/D2/D3)
-0x0054 - 84 IEEEDPDiv(Double, Double)(D0/D1/D2/D3)
-0x005A - 90 IEEEDPFloor(Double)(D0/D1)
-0x0060 - 96 IEEEDPCeil(Double)(D0/D1)

mathieeedoubtrans.library

-0x001E - 30 IEEEDPAtan(Double)(D0/D1)
-0x0024 - 36 IEEEDPSin(Double)(D0/D1)
-0x002A - 42 IEEEDPCos(Double)(D0/D1)
-0x0030 - 48 IEEEDPTan(Double)(D0/D1)
-0x0036 - 54 IEEEDPSincos(Double, Pf2)(A0,D0/D1)
-0x003C - 60 IEEEDPSinh(Double)(D0/D1)
-0x0042 - 66 IEEEDPCosh(Double)(D0/D1)
-0x0048 - 72 IEEEDPTanh(Double)(D0/D1)
-0x004E - 78 IEEEDPExp(Double)(D0/D1)
-0x0054 - 84 IEEEDPLog(Double)(D0/D1)
-0x005A - 90 IEEEDPPow(Exp, Arg)(D2/D3,D0/D1)
-0x0060 - 96 IEEEDPSqrt(Double)(D0/D1)
-0x0066 - 102 IEEEDPTieee(Double)(D0/D1)
-0x006C - 108 IEEEDPFieee(Single)(D0)
-0x0072 - 114 IEEEDPAsin(Double)(D0/D1)
-0x0078 - 120 IEEEDPAcos(Double)(D0/D1)
-0x007E - 126 IEEEDPLog10(Double)(D0/D1)

mathtrans.library

-0x001E - 30 SPAtan(Float)(D0)
-0x0024 - 36 SPSin(Float)(D0)
-0x002A - 42 SPCos(Float)(D0)
-0x0030 - 48 SPTan(Float)(D0)
-0x0036 - 54 SPSincos(LeftFloat, RightFloat)(D1,D0)
-0x003C - 60 SPSinh(Float)(D0)
-0x0042 - 66 SPCosh(Float)(D0)
-0x0048 - 72 SPTanh(Float)(D0)
-0x004E - 78 SPExp(Float)(D0)
-0x0054 - 84 SPLog(Float)(D0)
-0x005A - 90 SPPow(LeftFloat, RightFloat)(D1,D0)
-0x0060 - 96 SPSqrt(Float)(D0)
-0x0066 - 102 SPTieee(Float)(D0)
-0x006C - 108 SPFieee(Integer)(D0)
-0x0072 - 114 SPAsin(Float)(D0)
-0x0078 - 120 SPAcos(Float)(D0)
-0x007E - 126 SPLog10(Float)(D0)

translator.library

-0x001E - 30 Translate(InputString, InputLength, OutputBuffer, BufferSize)
(A0,D0/A1,D1)

Anhang D: Systemnahe BASIC-Befehle

Für den BASIC-Programmierer, der wie wir mit den Libraries arbeitet, ist es wichtig zu wissen, welche BASIC-Befehle ihn bei der täglichen Arbeit unterstützen. Es gibt davon einige, die allesamt wichtige Aufgaben erfüllen. Hier ist deshalb noch einmal in einer Zusammenfassung eine Liste der Befehle mit Format, Parametern, Ergebnis und Anwendung.

Sie können hier nachschlagen, wenn Sie genau wissen, daß es sich um einen BASIC-Befehl handeln muß, der im System "rumwühlt", von dem Sie aber nur seine Funktion - nicht seinen Namen kennen.

AmigaBASIC

ASC(String$) - Tabellenwert des Zeichens ermitteln

Die ASC()-Funktion ermittelt von einem String, den ASCII-Wert, der ihm nach einer genormten Tabelle zugeordnet wird. So hat das A den Wert von 65, und alle weiteren Buchstaben entsprechend ihrer Position im Alphabet, höhere Werte. Die Ziffer 0 steht in der Tabelle an 48. Stelle, und so geht es mit anderen Zeichen weiter. Wenn Sie die Werte für alle möglichen Zeichen wissen wollen, sollten Sie den umgekehrten Weg mit der CHR$()-Funktion gehen, der man einen Wert zwischen 0 und 255 übergibt und die dafür auf der anderen Seite der Tabelle nachsieht, welches Zeichen dem Wert zugeordnet wird.

Benötigt wird diese Funktion beim Schreiben von Texten in bestimmte Speicherbereiche. Weil die POKE-Anweisung (siehe weiter unten) nicht in der Lage ist, Zeichen an eine Speicherstelle zu schreiben, nimmt man einfach den ASCII-Wert, der dem Zeichen entspricht.

PEEK(Adresse) - 1-Byte-Wert einer Speicherstelle ermitteln

Mit der PEEK()-Funktion ermittelt man unter Angabe einer Speicheradresse den in ihr befindlichen Wert. Je nachdem, was man in der Speicherstelle erwartet, muß der Wert danach ausgewertet werden. Handelt es sich um die

Daten einer Struktur, stehen an bestimmten Positionen im Speicher wichtige Daten, die nur noch ausgelesen werden müssen.

PEEKW(Adresse) - 2-Byte-Wert einer Speicherstelle ermitteln

Die PEEKW()-Funktion arbeitet genauso wie die PEEK()-Funktion mit dem Unterschied, daß bei PEEK() nur ein Byte aus dem Speicher ausgelesen wird. Mit PEEKW() werden zwei Bytes hintereinander ausgelesen und verknüpft. Wenn sich mit einem Byte nur 256 verschiede Zustände darstellen lassen, ist mit zwei mal 256 Zuständen schon ein Bereich von 65536 Zahlen abzudecken. Dies wird mit dieser Funktion vorzeichenrichtig gemacht.

PEEKL(Adresse) - 4-Byte-Wert einer Speicherstelle ermitteln

Genauso wie auch PEEKW() arbeitet PEEKL(), mit dem erneuten Unterschied, daß sich wiederum die Breite der Funktion verdoppelt. Es können jetzt nicht zwei zusammenhängende Bytes ausgelesen werden, sondern vier. Dies wird für sehr große Zahlenwerte benötigt und übergreift 4294967296 Zustände.

POKE Adresse, Wert
- 1-Byte-Wert in eine Speicherstelle schreiben

Im Gegensatz zu den PEEK()-Funktionen sind die POKE-Befehle zum Schreiben in den Speicher vorgesehen. Auch hier wird die Speicheradresse, auf die zugegriffen werden soll, angegeben. Allerdings wird noch ein weiterer Wert benötigt, nämlich der, der dort hineingeschrieben werden soll. Er darf je nach POKE-Befehl nur in einem bestimmten Wertebereich liegen. Beim einfachen POKE ist dies von 0 bis 255.

POKEW Adresse, Wert
- 2-Byte-Wert in eine Speicherstelle schreiben

Der POKEW-Befehl arbeitet genau wie auch der normale POKE-Befehl. Der Wertebereich und der Zugriff sind allerdings größer. Es können Zahlen von 0 bis 32767 geschrieben werden. Dies erfolgt allerdings in zwei Bytes.

POKEL Adresse, Wert
- 4-Byte-Wert in eine Speicherstelle schreiben

Auch POKEL schreibt Zahlenwerte in den Speicher. Dafür verwendet es eine Breite von 4 Bytes, in die Werte von 0 bis 4294967295 eingetragen werden können.

SADD(String$) - Speicherstelle des Strings ermitteln

Mit dieser Funktion wird die Startadresse eines Strings ermittelt. Das heißt, man bekommt die Adresse, an der das erste Zeichen im Speicher steht.

VARPTR(Variable) - Speicherstelle der Variablen ermitteln

Diese Funktion arbeitet ähnlich wie SADD(), allerdings wird hier die Adresse einer Zahl-Variablen ermittelt, die sich in der Position von der einer String-Variablen unterscheidet. Auch hier bekommen wir die Adressse, in der die Werte gespeichert werden.

GFA-BASIC

ARRPTR(Feld()) - Speicherstelle des Variablenfeldes ermitteln

Diese Funktion ermittelt die Speicheradresse, an der das erste Element eines Variablen-Feldes untergebracht ist. Diese Funktion arbeitet ähnlich wie VARPTR().

CVD(String) - Zeichenkette in 8-Byte-Fließkommazahl

Wandelt eine 8 Zeichen lange Zeichenkette in die entsprechend lange Fließkommazahl. Die Funktion stellt die Umkehr von MKD$() dar.

CVI(String) - Zeichenkette in 2-Byte-Integer

Wandelt eine 2 Zeichen lange Zeichenkette in die entsprechende 2-Byte-Integer-Zahl. Die Funktion stellt die Umkehr von MKI$() dar.

CVL(String) - Zeichenkette in 4-Byte-Integer

Wandelt eine 4 Zeichen lange Zeichenkette in die entsprechende 4-Byte-Integer-Zahl. Die Funktion stellt die Umkehr von MKL$() dar.

CVS(String)
- Zeichenkette im Amiga-Format in 4-Byte-Fließkommanzahl

Wandelt eine 4 Zeichen lange Zeichenkette im AmigaBASIC-Format in die entsprechende Zahl im GFA-Format. Die Umkehrfunktion heißt MKS$().

DPEEK(Adresse) - Double-Peek

Siehe PEEKW in AmigaBASIC

DPOKE Adresse,Zahl - Double-Poke

Siehe POKEW in AmigaBASIC

LPEEK(Adresse) - Long-Peek

Siehe PEEKL in AmigaBASIC

LPOKE Adresse,Zahl - Long-Poke

Siehe POKEL in AmigaBASIC

MKD$(Zahl) - Make Double to String

Wandelt eine 8-Byte-Fließkommazahl in den entsprechenden String um. Das heißt, daß jedes Byte in den entsprechenden Character-String gesetzt wird. 8 Bytes ergeben den gesamten String. Siehe dazu auch CVD().

MKI$(Zahl) - Make Integer to String

Wandelt eine 2-Byte-Integer-Zahl in den entsprechenden String. Siehe weitere Daten oben. Siehe dazu auch CVI().

MKL$(Zahl) - Make Long to String

Wandelt eine 4-Byte-Integer-Zahl (long Integer) in den entsprechenden String. Siehe weitere Daten unter MKD$(). Siehe dazu auch CVL().

MKS$() - Make Float to AmigaBASIC-String

Wandelt eine 4-Byte-Fließkommazahl in die entsprechende Zeichenkette im AmigaBASIC-Format, so daß Daten unter den Programmiersprachen ausgetauscht werden könne. Siehe dazu auch CVS().

Anhang E: Dankwort an die hilfeleistenden Firmen

Im folgenden möchten wir den Firmen danken, die uns bei der Erstellung dieses Buches hilfreich mit Hard- und Software zu Seite gestanden haben.

Firma
GIGATRON (Hersteller)

Resthauser Str. 1
4590 Cloppenburg
Tel. 04471/3070

FreeCom (Vertrieb)

Bismarckstraße 2
2000 Hamburg 20
Tel. 040/495990

für die Speichererweiterung "MiniMax", das nette Telefongespräch und die Informationen zu den Verhältnissen auf dem Speicher-Markt

Firma
Kupke Computertechnik GmbH

Burgweg 52a
4600 Dortmund 1

für die Diskettenlaufwerke

Firma
ALCOMP

Lessing Str. 46
5012 Bedburg

für Kickstart-Umschaltung, Trackdiskanzeige und Bootselector

Firma
RUSHWARE GmbH

Bruchweg 128-132
4044 Kaarst 2

für die Software (The Graphics Studio, DeluxeMusic ConstructionSet, Instant Music) und die freundliche Telefon-Beratung von Susanne Dieck

Firma
Message Computer

Stöckmannstr. 78
4200 Oberhausen

für Kickstart-Umschaltung, Speichererweiterungen und Festplatte

Firma
COMBITEC Computer GmbH

Liegnitzer Str. 6 - 6a
5810 Witten

für die Bereitstellung der Festplatte mit 500er-Adapter

Firma
CompTec

Obermörmterer Straße 1
4192 Kalkar 4

für den Bausatz zum Umbau des Amiga 500 zum 2000er

Firma
CompuStore

Fritz-Reuter-Straße 6
6000 Frankfurt/Main 1
Tel. 069/567399

für die freundliche Bereitstellung der Software "Calligrapher", "CityDesk" und "Superbase" und natürlich auch der

Firma
DATA BECKER GmbH

Merowingerstr. 30
4000 Düsseldorf 1
Tel. 02 11/31 00 10

für die Bereitstellung von Hard- und Software

Firma
vortex

Falterstraße 51-53
7101 Flein bei Heilbronn
Tel. 0 71 31/50 88 - 0

für die preiswerte Festplatte

Firma
Soyka Datentechnik GmbH

Hattinger Straße 685
4630 Bochum 5
Tel. 02 34/4 98 25 - 26

für den Digitizer "Digi View Gold"

Firma
DTM

Poststraße 25
6200 Wiesbaden-Bierstadt
Tel. 0 61 21/56 00 84

für den Diskettenmonitor DISCovery

Wir hoffen auf weiterhin gute Zusammenarbeit.

Stichwortverzeichnis

[Amiga]-Tasten ... 483
[Backspace] ... 484
[Del] ... 484
[Enter]-Taste ... 484
[Return]-Taste ... 483, 484
[Shift]-Tasten ... 483
5¼-Zoll-Laufwerke ... 213
68000-Prozessor ... 489
68010-Prozessor ... 18

Aegis Sonix ... 164
Alert ... 520
Amiga-Libraries ... 287
AmigaBASIC ... 15
AmigaDOS ... 520
Apfelmännchen ... 276
ASCII ... 521
Assembler ... 223, 380, 521
Audiomaster II ... 173
AUX ... 60

BASIC ... 275, 522
Baud ... 256
Benutzeroberfläche ... 522
Betriebssystem ... 523
Binäres Zahlensystem ... 523
BindDrivers ... 19
Bit ... 523
Blitter ... 490
BlocksPerTrack ... 227
Booten ... 523
BSS ... 433
Buffers ... 228
BufMemTyp ... 229

Button ... 524
Byte ... 524

C ... 524
CD-ROM ... 221
CLI ... 501, 524
Co-Prozessoren ... 490
Code ... 433
Code-Segment ... 433
Compiler ... 525
CON ... 21
Copper ... 490
Cursor ... 525
Custom-Chips ... 490

Data ... 433
DATAMAT ... 138
Datei ... 525
Datenverwaltung ... 526
Datex-P ... 251
Datex-P-Teilnehmerkennung ... 251
Denise ... 490
DeleteLayers() ... 18
Deluxe Paint ... 146
DeluxeMusic ... 168
Device ... 226, 506, 526
DEVS ... 42
Dezimalsystem ... 49
DFÜ ... 254
DH0 ... 59
Digitalisierer ... 243
Directory ... 526
Disk-Monitor ... 337
DiskDoctor ... 224
Diskette ... 526
Diskettenlaufwerk ... 495
DMA-Logik ... 490
DOS ... 510, 527
DosType ... 229
Double ... 445

Drucker ... 497
Druckertreiber ... 26, 27
Dualsystem ... 49, 527

Echt-Zeit ... 164, 193
Editor ... 441, 527
Empty ... 36
EVEN CIA ... 203
Expansion ... 48

Farbmonitor ... 487
FastFileSystem ... 224
FastFonts ... 19
Fat Agnus ... 490
FD-File ... 285
Festplatte ... 59, 218, 528
FF ... 19
FFS ... 224
File ... 528
File-Request-Box ... 319
FileSystem ... 226
Flags ... 227
Float ... 445
Floppy ... 275, 495 529
FONTS ... 515, 529

Gadget ... 529
Gary ... 203, 490
Gate-Array-Chip ... 490
GFA-BASIC ... 357
GlobVec ... 229
Grafik ... 275
Grafik-Programmierung ... 363
Graphics Studio ... 153
Guru-Meditation ... 18, 529

Handler ... 507
Hardware ... 14, 530
Hexadezimalsystem ... 50
HighCyl ... 228

I/O ... 492
Icon ... 530
IFF ... 145, 530
Int ... 445
Interleave ... 228
Interpreter ... 531
Intuition ... 531

Joker ... 80
Joystick ... 531
Joystick-Weiche ... 270

Keymaps ... 514
Kickstart ... 259, 531
Kickstart-ROM ... 203
Kickstart-Umschaltung ... 259
Koppler ... 253

Laser-Drucker ... 498
Libraries ... 15, 504, 532
Library-Funktionen ... 282
Long ... 445
Long float ... 445
Long int ... 445
LowCyl ... 228

Maschinensprache ... 532
Massenspeicher ... 495
Maus ... 485, 532
MC68010 ... 266
Menü ... 533
Modem ... 251
Motorola ... 489
Mount ... 229
MountList ... 225
MS-DOS ... 221, 465
Multitasking ... 533
Musik ... 162

NEWCON 21
Notepad 28
Nullmodem 253

ODD CIA 203

PAR 59
Partition 224, 225
Patch 534
Paula 490
PC 15, 465
PC-Karte 221, 469
PC-Platine 465
Prefs 44
Priority 230
Programmiersprache 379, 534
Prozessor 380, 535
PRT 59
Quellcode 433

RAD 109
RAM 22, 535
RAM-Chips 199
RAM-Disk 536
Read-Error 223
Requester 536
Reserved 228
Reset 536
Resident 101
ROM 203, 279, 537
RS232 253

Screen 537
SCSI 221
SCSI-Karte 221
SCSI-Platten 221
SCSI-Schnittstelle 221
SER 59
Shell 41
SideCar 469

Software ... 537
Speichererweiterungen ... 193
Speicher ... 491
Speicherverwaltung ... 291, 367
StackSize ... 230
Standard ... 538
Startup-Sequence ... 17
Superbase ... 181
Surfaces ... 227
System ... 45
Systemprogrammierung ... 275

Tastatur ... 482, 538
Textattribute ... 317
TEXTOMAT ... 125
Textstil ... 317
Topaz ... 515
Trashcan ... 36
TURBOprint ... 185

UBM ... 125
Unit ... 227
Utilities ... 46

Verschachtelungstiefe ... 361
Verschieben ... 73
Verschiedene Zeichensätze ... 314
Verzeichnis ... 61, 539
Viren ... 105, 539
Vorformatieren ... 224

Wait-States ... 491
Window ... 539
Workbench ... 499, 540

Zeichensatz ... 540
Zeileneditor ... 540
Ziffernblock ... 484

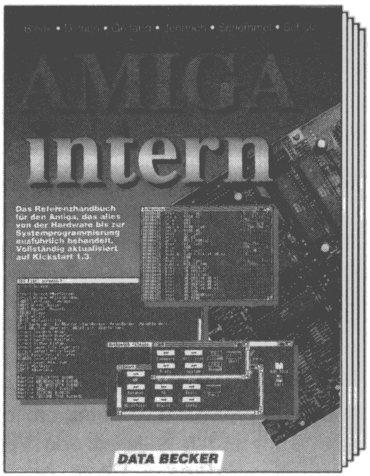
Die ganze, farbenfrohe Palette von DPaint III.

DPaint III gehört wohl zu den außergewöhnlichsten Grafikprogrammen, die derzeit für den Amiga verfügbar sind. Bereits mit den einfachen, „normalen" Funktionen erzielt man erstaunliche Ergebnisse. Mit dem großen DPaint-III-Buch jedoch wird's erst so richtig professionell. Neben einer ausführlichen, leichtverständlichen Beschreibung der DPaint-Grundfunktionen zeigt dieses Buch vor allem, was DPaint tatsächlich leisten kann: perspektivische Zeichnungen, verbogene Brushs, Animation in 3D, Erstellen von Videos... Dazu alles Wichtige über den Datenaustausch mit anderen Programmen. Natürlich verraten Ihnen die Autoren auch ihre zahlreichen Tips und Tricks, mit denen sie noch mehr aus diesem Programm herausholen. Ein hilfreicher Anhang, der alle Funktionen übersichtlich darstellt, rundet das Ganze ab und macht diesen Band zu einem Standardwerk, in dem auch der Profi immer wieder nachschlagen wird. Das große DPaint-III-Buch – für alle, die nach Höherem streben.

Langlotz/Vignjevic
Das große DPaint-III-Buch
393 Seiten, DM 39,-
ISBN 3-89011-369-9

In einem Band: alle Details zu Ihrem Amiga.

Amiga Intern – von der ersten bis zur letzten Seite bietet Ihnen dieses einzigartige Nachschlagewerk alle harten Facts zu Ihrem Rechner. Angefangen von einer detaillierten Beschreibung des 68000-Prozessors, der CIA, der Custom-Chips und der Schnittstellen über die Hardware-Programmierung bis hin zu einer leichtverständlichen Dokumentation aller Library-Funktionen – zu allen bisher ausgelieferten Kickstart-Versionen. Aus dem Inhalt: die Strukturen von EXEC, I/O-Handhabung, Verwaltung der Resources, Erstellen eigener Devices, EXEC-Base, resetfeste Programme, DOS-Funktionen, interne DOS-Bibliothek, Aufbau einer Diskette, Autoboot mit der ROMboot.library, Programmierung eigener Handler, Ein- und Ausgabe über die verschiedenen Amiga-Devices, Standard-Austausch-Formate und IFF-Komprimierungsverfahren, alle Amiga-Libraries mit den dazugehörigen Strukturen, Basis- und Grundstrukturen, Preferences als Datenstruktur, Datenübermittlung von Workbench und CLI, Konventionen im Programmierstil...



Bleek/Dittrich/Gelfand/Jennrich/Schemmel/Schulz
Amiga Intern
Hardcover, 1095 Seiten, DM 98,-
ISBN 3-89011-398-2

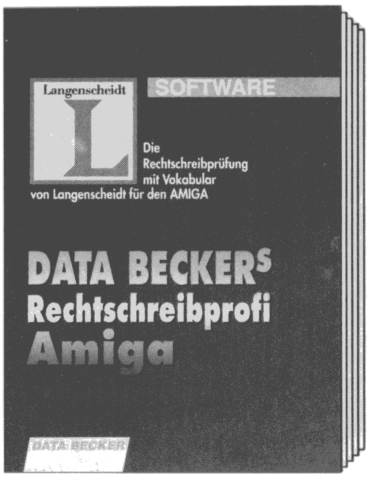
Kompetentes Detailwissen zum Amiga 2000.

Von den glorreichen Drei (Rügheimer/Spanik/Amiga) wurden die Amiga-Anwender schon immer verwöhnt: Ihre Fachbücher strotzen vor fundiertem Know-how und sind dennoch dank einer lockeren Schreibe überaus unterhaltsam. Warum sollte es also beim großen Amiga-2000-Buch anders sein? Mit viel Liebe zum Detail und der gewohnten leichtverständlichen Sprache beschreiben Sie alles, was den Amiga 2000 so interessant macht: die möglichen Speichererweiterungen, Einbau und Einrichtung einer PC-/Amiga-Harddisk, Arbeiten mit einer PC-/AT-Karte, Kickstart im RAM, jede Menge zur Janus.Library 2.0, Software-Installationstips, der Umgang mit dem AmigaDOS und und und. Selbstverständlich auch mit einer detaillierten Einführung zum Amiga 2000 und einigen wichtigen Software-Tips. Das große Amiga-2000-Buch – ein großartiges Buch zu einem großartigen Rechner.

Rügheimer/Spanik
Das große Amiga-2000-Buch
Hardcover, 736 Seiten, DM 59,-
ISBN 3-89011-199-8

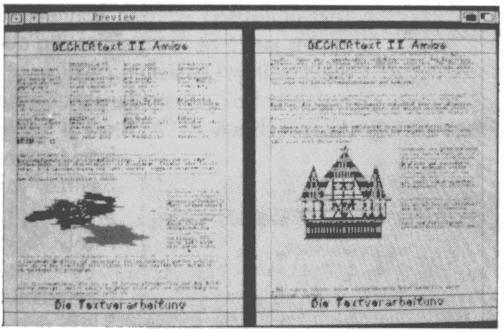
Alles korrekt? Fragen Sie den Rechtschreibprofi.

DATA BECKERs Rechtschreibprofi ist das schnelle Prüfprogramm für alle, die Wert auf einwandfreie Texte legen. Ganz gleich, welche Textverarbeitung Sie besitzen: Lassen Sie Ihre Berichte, Artikel, Protokolle, wissenschaftliche Texte und Diplom- oder Studienarbeiten bequem vom Computer „gegenlesen". Die komfortable Oberfläche erlaubt auch Computer-Neulingen die rasche Nutzung des Programms. Für die Richtigkeit des Prüfvorgangs garantiert ein bekannter Name: Langenscheidt-Mitarbeiter haben die rund 106.000 Wörter und Flexionen des Hauptwörterbuchs zusammengestellt. Da DATA BECKERs Rechtschreibprofi sich am Wortstamm orientiert, sind damit auch alle möglichen Kombinationen erfaßt: Insgesamt werden also 2,5 Millionen Wörter überprüft. Natürlich haben Sie außerdem die Möglichkeit, weitere Fachausdrücke und fremdsprachliche Begriffe im Benutzerlexikon abzulegen.



DATA BECKERs Rechtschreibprofi Amiga
ISBN 3-89011-585-3
DM 99,- (unverb. Preisempfehlung)

BECKERtext II Amiga: Komfort wie noch nie.



Keine andere Textverarbeitung war bei Amiga-Freunden und Fachredakteuren so beliebt wie BECKERtext. War? BECKERtext tritt als in vielen Tests ungeschlagener Champion ab – denn jetzt feiert BECKERtext II eine glanzvolle Premiere.

Lesen Sie auf den folgenden beiden Seiten, welche Standards BECKERtext II für Amiga-Programme setzt – und warum die neue Textverarbeitung natürlich der ideale Partner für DATA BECKERs Rechtschreibprofi ist.

Textverarbeitung, daß die Augen vor Staunen übergehen? Das kann nur das „What You See Is What You Get" von BECKERtext II sein. Bis zu sechs frei einstellbare Spalten pro Seite; sämtliche Amiga-Fonts (bis 48 Punkt!) auf Bildschirm und Drucker (auch proportional); nachträgliches Verschieben, Verzerren, Vergrößern und Verkleinern von eingebundenen Grafiken (die rechts oder links von Text umflossen werden können); Seitenvorschau mit bis zu 15 Seiten gleichzeitig auf dem Monitor; gleichzeitige Darstellung mehrerer, beliebig angeordneter Dokumentenfenster: Komfort, wohin das Auge schaut. BECKERtext II Amiga bietet viele Funktionen, die Amiga-Besitzer bisher nur von Profi-Programmen aus der MS-DOS-Welt kannten.

Noch schöner: BECKERtext II läßt sich mit frei konfigurierbaren Menüs, einer beliebig belegbaren Tastatur und selbst geschriebenen Programmen (Makros, ARexx-Schnittstelle) zu einer ganz individuellen Textverarbeitung formen. Definieren Sie Ihre eigenen Shortcuts, arbeiten Sie also ohne Maus, wenn Ihnen der Sinn danach steht: Alle Befehle können über die Tastatur eingegeben werden.

BECKERtext II kommt Anfängern genauso gelegen wie erfahrenen Anwendern. Die einen nutzen den speziellen Kurs für Einsteiger im Handbuch, die anderen profitieren von der bequemen Index-Erstellung, dem leistungsfähigen Gliederungseditor oder den vielen Druckoptionen. Auf der rechten Seite finden Sie die wichtigsten Funktionen von BECKERtext II auf einen Blick. Am besten schauen Sie aber zweimal hin – es lohnt sich.

BECKERtext II Amiga
unverb. Preisempfehlung DM 298,-
ISBN 3-89011-584-5

Alles auf einen Blick:

WYSIWYG:
- Textverarbeitung wahlweise im „Normal-Modus" oder im „WYSIWYG-Modus" mit allen Editierfunktionen und mit der Anzeige aller Zeichen- und Absatzformate, Seitenformate und Seitenränder, mit bis zu sechs Spalten, Kopf- und Fußzeilen, Fußnoten, Grafiken etc.
- Ganzseitendarstellung von bis zu 15 Seiten gleichzeitig, mit Seitennummern-Angabe, Vor- und Zurückblättern, Cursorpositionierung für gezielte Rückkehr zum Text
- Gleichzeitige Darstellung mehrerer Dokumentenfenster, deren Anordnung frei definierbar ist

Bedienung:
- Pulldown-Menüs und Mausbedienung
- sämtliche Befehle über Tastatur und mit Maus aktivierbar
- Shortcuts sind frei belegbar
- Menüs/Untermenüs sind frei belegbar und können selbst konfiguriert werden
- freie Tastaturbelegung (Makros, Floskeln, Funktionen)
- übersichtliche Dateiauswahlbox mit direkt anwählbaren Symbolen für alle angeschlossenen Geräte
- Lineal mit Tabulatorleiste
- Kommandozeile zur direkten Eingabe von Befehlen
- UNDO/REDO
- Automatische Silbentrennung
- AREXX-Schnittstelle zu Fremdprogrammen, voll programmierbar

Druck:
- Formatschablonen für Zeichen-, Absatz- und Bereichsformate, Dokument
- Separates Speichern der Formatschablonen
- Hinzuladen einer separat gespeicherten Formatschablone
- Anzeige des aktuellen Druckformats
- Drucken von selektierten Seiten
- Drucken im Hintergrund
- Druck ist jederzeit abbrechbar
- Drucken im Hoch- oder Querformat, im Text- oder Grafikmodus

Grafiken:
- bis zu 16 Farben
- Grafiken können im Text frei verschoben, vergrößert, verkleinert und verzerrt werden
- Grafiken können mit Rahmen versehen und rechts/links von Text umflossen werden
- Zusatzprogramm BT-SNAP zum Ausschneiden/Fotografieren beliebiger Bildschirmausschnitte

Serienbrief:
- Befehle für einmalige Abfrage
- Befehle für Abfrage pro Brief
- Steuerdateien

Index:
- Kennzeichnung durch Indexmarker für automatisches Erstellen
- Anlegen von verschachtelten Indizes
- selbständiges Entfernen von Mehrfacheinträgen

Kopf-/Fußzeilen:
- Abstände von oben/unten frei wählbar
- können sowohl Text als auch Grafiken enthalten

Fuß-/Endnoten:
- automatische oder manuelle Wahl des Fuß-/Endnotenzeichens

Mehrspaltensatz:
- Darstellung von bis zu sechs Spalten
- Spaltengrößen frei einstellbar

Tabulatoren:
- Typen: linksbündig, rechtsbündig, zentriert, dezimal
- Füllzeichen frei wählbar

Gliederungseditor:
- beliebige Verschachtelungen und Hierarchien möglich
- Auf- und Zuklappen von verschiedenen Gliederungsebenen

Rechenfunktionen:
- Rechnen im Text mit den vier Grundrechenarten und Prozent
- Textstatistik über Anzahl der Wörter, Zeilen und Absätze
- Anzeige des noch freien Speichers

Masken:
- Einfaches Erstellen von Formularen über definierte Platzhalter

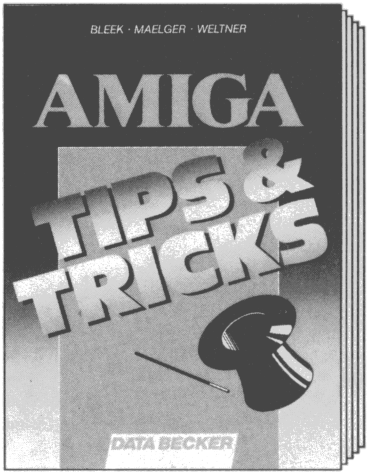
Rechtschreibprüfung:
- Rechtschreibprüfung aus DATA BECKERs Rechtschreibprofi integrierbar

Mindestkonfiguration:
- Amiga 500 mit 1MB Speicher

Der einfache Einstieg:
- BECKERtext II Amiga wird mit einer ausführlichen Beschreibung ausgeliefert. Der Band enthält einen speziellen Kurs für Einsteiger mit grundlegenden Informationen zum Amiga und zur Textverarbeitung.

Wer die richtigen Kniffe kennt, ist erfolgreicher.

Zu nahezu jedem Arbeitsbereich gibt es spezielle Tricks, mit denen man noch schneller und vor allem effektiver arbeiten kann. Tricks, auf die Profis bei ihrer täglichen Arbeit stießen und die daher nur selten in den entsprechenden Dokumentationen zu finden sind. In „Amiga – Tips & Tricks" verraten Ihnen Amiga-Experten, welche Kniffe sie sich in ihrer langjährigen Arbeit mit dem Amiga angeeignet haben. Zu allen Anwendungsbereichen und unter Berücksichtigung des Betriebssystems 1.3: Gestaltung eigener Programme, Tips & Tricks zum AmigaBASIC, Einsatz von DOS-Routinen, Optimierung für AmigaBASIC-Programme, Tips zur Arbeit mit der Workbench, Icon-Aufbau, die neuen Preferences, Nutzung der CLI-Befehle und Devices. Arbeiten Sie wie die Profis. Mit diesem Buch haben Sie das Zeug dazu.



Bleek/Maelger/Weltner
Amiga – Tips & Tricks
Hardcover, 555 Seiten, DM 49,-
ISBN 3-89011-211-0

3. erweiterte und überarbeitete Auflage

Das große Amiga-500-Buch

Der Amiga 500 wird immer beliebter. Kunststück, hat er doch wirklich das Zeug zum Traumcomputer. Besonders für diejenigen, die einen Arbeitsrechner und einen Freizeitcomputer wollen. Das einzige, was allen Amiga-500-Besitzern noch fehlte, ist die entsprechende Literatur, die alles in und um den Amiga 500 herum behandelt. Lernen Sie Ihren Amiga 500 einmal von einer ganz anderen Seite kennen, und Sie werden feststellen, daß dieser Computer nicht nur zum Spielen, sondern auch ideal für ernsthafte Anwendungen geeignet ist.

In diesem Buch finden Sie zu beinahe allen Bereichen praktische Tips & Tricks. Sei es für den Workbench- oder CLI-Benutzer, den Programmierer oder den Anwender. Dabei lernen Sie nicht nur den Umgang mit CLI und Workbench, sondern erfahren auch etwas über die Systemprogrammierung, den professionellen Einsatz von Software und die wichtigsten Hardwareerweiterungen. Ob Sie Einsteiger oder Profi sind, in diesem Buch finden Sie alle wichtigen Informationen rund um Ihren Amiga 500.

Besser — Damit Sie von Anfang an gleich richtig loslegen können: Richtiger Einsatz der mitgelieferten Programme mit vielen Tips & Tricks zum CLI/Shell • Nutzung des Betriebssystems von AmigaBASIC und GFA-BASIC • Intuition-Programmierung unter BASIC am Beispiel eines Diskettenmonitors.

Schneller — Sie wollen mehr Rechenleistung zu einem günstigen Preis?! Dann rüsten Sie Ihren Amiga mit dem Einbau einer 68010 CPU auf.

Aktueller — Nutzen Sie die Möglichkeit, aus der RAM-Disk oder von der Festplatte zu booten - vom Einbau der Umschaltplatine, der Installation bis zum optimalen Einsatz der Harddisk.

Praktischer — Ob Sie das SideCar oder die PC-Karte des Amiga 2000 einsetzen wollen, bleibt Ihnen überlassen, aber auf dem Weg zum PC sollten sich sich nicht auf Ihr Glück verlassen, sondern das Praxis-Know-how dieses Buches nutzen.

ISB N 3-89011-279-X DM +049.00

DM 49,–
ÖS 382,–
sFr 47,–

DATA BECKER

04900

9 783890 112794

This was brought to you

from the archives of

http://retro-commodore.eu